



PALAVRAS CHAVES/KEY WORDS
COMPUTAÇÃO GRÁFICA - PAC
MODELADOR GEOMÉTRICO
ESTRUTURA DE DADOS HIERÁRQUICA

AUTORES
AUTHORS

AUTORIZADA POR/AUTHORIZED BY
Ralf Gielow
Ralf Gielow
Pres. Cor. Pós-Graduação

AUTOR RESPONSÁVEL
RESPONSIBLE AUTHOR
Andrés F.P. Horna
Andrés F.P. Horna

DISTRIBUIÇÃO/DISTRIBUTION
 INTERNA / INTERNAL
 EXTERNA / EXTERNAL
 RESTRITA / RESTRICTED

REVISADA POR/REVISED BY
Otávio S.C. Durão
Otávio S.C. Durão

CDU/UDC
519.674

DATA / DATE
setembro/1990

TÍTULO/TITLE	PUBLICAÇÃO Nº PUBLICACION NO INPE-5118-TDL/420
	DESENVOLVIMENTO DE UM BANCO DE DADOS INTERATIVO COM ELEMENTOS GERADOS POR VARREDURA PARA UM SISTEMA CAD
AUTORES/AUTHORSHIP	Andrés Fernando Paredes Horna

ORIGEM
ORIGIN
PG/LAC

PROJETO
PROJECT
CAP

Nº DE PAG. NO OF PAGES 366	ULTIMA PAG. LAST PAGE A.9
VERSÃO VERSION	Nº DE MAPAS NO OF MAPS

RESUMO - NOTAS / ABSTRACT - NOTES

Este trabalho descreve a implantação de um software interativo que implementa um modelador geométrico com características hierárquicas para a construção de sólidos, em um sistema CAD. Neste trabalho os sólidos são construídos seguindo uma estrutura em árvore binária hierárquica cujos elementos de nível inferior (figuras bidimensionais ou primitivas básicas) são usados para a geração de sólidos através da técnica conhecida como varredura ("sweeping") rotacional ou translacional, em torno de um eixo de rotação ou ao longo de uma direção, respectivamente. Os sólidos gerados por varredura são então compostos entre si, usando operações booleanas, para a obtenção da forma final desejada. Inicialmente justifica-se a escolha do tipo de estrutura de banco de dados utilizado e se descreve seu funcionamento; em seguida é apresentada uma base teórica sobre modelamento geométrico e principais esquemas de representação de sólidos; finalmente, com o objetivo de viabilizar a transportabilidade do modelador geométrico a outros sistemas, usando pacotes gráficos padronizados, faz-se uma comparação com outros padrões.

OBSERVAÇÕES / REMARKS

Dissertação de mestrado em Computação Aplicada, aprovada em 13 de Março de 1990.

Aprovada pela Banca Examinadora
em cumprimento a requisito exigido
para a obtenção do Título de Mestre
em Computação Aplicada

Dr. Gerald Jean Francis Banon



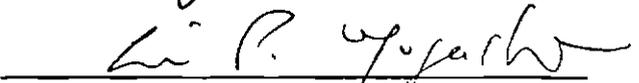
Presidente

Dr. Otávio Santos Cupertino Durão



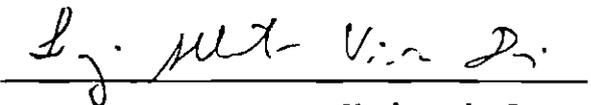
Orientador

Dr. Léo Pini Magalhães



Membro da Banca
-convidado-

Dr. Luiz Alberto Vieira Dias



Membro da Banca

Candidato: Andres Fernando Paredes Horna

são José dos Campos, 13 de março de 1990

ABSTRACT

This work presents the development of an interactive software that implements a geometrical modeller with hierarchical features for the construction of solids in a CAD system. In this work the solids are built following a hierarchical binary tree structure whose lowest level elements are used for generating solids through the technique known as rotational or translational "sweeping", either around a revolution axis or along a straight line, respectively. The swept generated solids are then composed with each other, using boolean operations, in order to obtain the final desired form. First, the kind of data base structure used is justified and its operation described; then a geometric modelling theoretical basis is presented as well as the main solid representation schemes. Finally, with the purpose of increasing the feasibility of the transportability of the geometric modeller to other systems, using standard graphic packages, a comparison with the most used graphic standards is made.

AGRADECIMENTOS

Ao Instituto de Pesquisas Espaciais INPE de São José dos Campos, pela oportunidade oferecida para a realização deste trabalho e para a obtenção do título de Mestre.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior CAPES e à IBM Brasil - Indústria, Máquinas e Serviços Ltda., pelos recursos financeiros concedidos durante o curso.

Ao Dr. Otávio Santos Cupertino Durão, pela idéia original do trabalho, seu incentivo, e à constante atenção e preocupação prestadas durante a orientação deste trabalho.

Aos doutores, membros da Banca Examinadora, pelos seus conselhos e críticas construtivas durante as avaliações do trabalho.

Ao Dr. Paulo Ouverá Simoni, por sua ajuda prestada como orientador acadêmico.

A minha família e a Jana, por todo o apoio prestado e por sua compreensão nas minhas horas de ausência.

Aos meus amigos, colegas de trabalho e outras pessoas, que de uma maneira direta ou indireta, colaboraram para a realização deste trabalho.

SUMÁRIO

	<u>Pág.</u>
<u>LISTA DE FIGURAS</u>	xxi
<u>CAPÍTULO 1 - INTRODUÇÃO</u>	1
<u>CAPÍTULO 2 - BANCO DE DADOS</u>	5
2.1 - Conceitos básicos	5
2.1.1 - Sistema de Banco de Dados SBD	5
2.1.1.1 - Dados	6
2.1.1.2 - Hardware	9
2.1.1.3 - Software	9
2.1.1.4 - Usuários	9
2.1.2 - A necessidade do banco de dados	10
2.1.3 - Estrutura de armazenamento	11
2.1.4 - Independência de dados	13
2.1.5 - Arquitetura de um SBD	14
2.2 - Estruturas de dados de um SBD	17

2.2.1 - Introdução	17
2.2.2 - Abordagem Relacional	18
2.2.3 - Abordagem Hierárquica	23
2.2.4 - Abordagem em Rede	27
2.3 - Conclusão e escolha de uma estrutura de banco de dados	32
2.4 - Proposta de uma estrutura de banco de dados gráfico	36
<u>CAPÍTULO 3 - MODELAMENTO GEOMÉTRICO</u>	53
3.1 - Introdução	53
3.1.1 - Definição de modelamento geométrico	53
3.1.2 - Formas geométricas nomeáveis e não nomeáveis .	55
3.1.3 - Equações paramétricas	55
3.2 - Curvas	59
3.2.1 - Definição	59
3.2.2 - Formas algébrica e geométrica	61
3.2.3 - Espaço objeto e espaço paramétrico da curva ..	64
3.2.4 - Linhas retas	65

3.2.5 - Curvas Spline	66
3.2.6 - Curvas de Bézier	66
3.2.7 - Curvas B-Spline	71
3.3 - Superfícies	77
3.3.1 - Definição	77
3.3.2 - Formas algébrica e geométrica	82
3.3.3 - Espaço paramétrico de uma superfície	87
3.3.4 - Superfície plana	89
3.3.5 - Superfície cilíndrica	91
3.3.6 - Superfície reguada	92
3.3.7 - Superfície de revolução	93
3.3.8 - Superfície esférica	97
3.3.9 - Superfície de Bézier	98
3.3.10 - Superfície B-Spline	100
3.4 - Sólidos	102
3.4.1 - Definições	103
3.4.2 - Forma algébrica e geométrica	107
3.4.3 - Vetores tangentes e vetores torção	110

3.4.4 - Espaço paramétrico do sólido	111
3.4.5 - Continuidade e sólidos compostos	112
3.5 - Transformações	114
3.5.1 - Transformações de modelamento	116
3.5.1.1 - Translação	117
3.5.1.2 - Rotação	118
3.5.1.3 - Mudança de escala	120
3.5.1.4 - Transformações inversas	121
3.5.1.5 - Composição	121
3.5.2 - Transformações de visualização	123
3.5.2.1 - Vista polar	123
3.5.2.2 - Vista de cena	127
3.5.3 - Transformações de projeção	128
3.5.3.1 - Projeção em perspectiva	129
3.5.3.2 - Projeção ortográfica (em paralelo)	131
3.6 - Fundamentos de modelamento de sólidos	133
3.7 - Esquemas para a representação de sólidos rígidos	141
3.7.1 - Introdução	141

3.7.2 - Modelos booleanos	142
3.7.3 - Decomposição celular	146
3.7.4 - Representação por varredura (representação "sweep")	147
3.7.5 - Sólido geométrico construtivo (CSG)	152
3.7.6 - Representação em arame (representação "wireframe")	156
<u>CAPÍTULO 4 - DESCRIÇÃO DO SISTEMA CAD UTILIZADO</u>	159
4.1 - Introdução	159
4.2 - Sistema CAD/CAM Computervision Designer V-X	160
4.2.1 - Hardware do sistema	161
4.2.2 - Software do sistema	163
4.2.3 - Nível OS	164
4.2.4 - Nível CADDS	165
4.2.4.1 - Modos model e draw	165
4.2.4.2 - Planos de construção	169
4.2.4.3 - Vistas	170
4.2.4.4 - "Layers"	172

4.2.4.5 - Entidades gráficas	173
<u>CAPÍTULO 5 - PADRONIZAÇÃO GRÁFICA</u>	175
5.1 - Introdução	175
5.2 - Fundamentos	175
5.3 - O processo da padronização	179
5.4 - Modelo de referência para a computação gráfica .	182
5.4.1 - Introdução	182
5.4.2 - O modelo de referência	183
5.4.2.1 - Visão de alto nível do modelo de referência	184
5.4.2.2 - Visão de baixo nível do modelo de referência	186
5.5 - O sistema CORE	190
5.5.1 - Introdução	190
5.5.2 - Interface DI/DD (Independente/Dependente de dispositivo)	192
5.5.3 - Estrutura de saída - Parte CORE independente de dispositivo	195
5.5.3.1 - Processo de visualização	196
5.5.3.2 - Transmissor	197

5.5.3.3 - Arquivo de "pseudo-display" PDF	198
5.5.3.4 - Simulações	199
5.5.4 - Estrutura da entrada - parte CORE independente de dispositivo	199
5.5.4.1 - Fila de eventos	200
5.5.4.2 - Associações	201
5.5.4.3 - Entrada síncrona/assíncrona	201
5.5.5 - Interação entre entrada/saída	203
5.5.6 - Tratamento de erros	203
5.5.7 - Estrutura dependente de dispositivo	203
5.6 - CGM - "Computer Graphics Metafile"	204
5.6.1 - Introdução	204
5.6.2 - Definição de metarquivo gráfico	205
5.6.3 - Relações de um metarquivo com um sistema gráfico	206
5.6.4 - Tipos de metarquivos	206
5.6.5 - Definição de CGM	209
5.6.6 - O que CGM padroniza	210
5.6.7 - Estrutura de CGM	212

5.6.8 - Aplicações de CGM	214
5.6.8.1 - Acesso de dispositivos gráficos redireciona- dos	215
5.6.8.2 - Armazenamento de desenhos	216
5.6.8.3 - Textos e gráficos misturados	217
5.7 - CGI - " Computer Graphics Virtual Device In- terface "	218
5.7.1 - Introdução	218
5.7.2 - História de CGI	220
5.7.3 - Estrutura de CGI	221
5.7.3.1 - Conceitos geométricos	221
5.7.3.2 - Controle de dispositivo	222
5.7.3.3 - Primitivas gráficas	223
5.7.3.4 - Armazenamento de primitivas	223
5.7.3.5 - Funções de entrada de CGI	225
5.7.3.6 - Ligações ("bindings")	226
5.7.3.7 - Tratamento de erros	226
5.8 - GKS - "Graphical Kernel System"	227
5.8.1 - Introdução	227

5.8.1.1 - Definição de pacote gráfico	228
5.8.1.2 - Requisitos de um pacote gráfico	228
5.8.2 - O GKS	230
5.8.3 - Estrutura do GKS	232
5.8.4 - Inicialização do GKS	235
5.8.5 - Primitivas gráficas de saída do GKS	236
5.8.5.1 - "Polyline"	236
5.8.5.2 - "Polymarker"	239
5.8.5.3 - "Text"	240
5.8.5.4 - Tratamento de cores	242
5.8.5.5 - "Cell array"	244
5.8.5.6 - "Fill area"	245
5.8.5.7 - GPD	247
5.8.6 - Sistemas de coordenadas	248
5.8.6.1 - Introdução	248
5.8.6.2 - Coordenadas normalizadas	249
5.8.7 - Segmentação	251
5.8.7.1 - Construção de segmentos	252

5.8.7.2 - Atributos de segmentos	253
5.8.8 - Dispositivos lógicos de entrada	255
5.8.8.1 - Modos de operação dos dispositivos lógicos de entrada	259
5.8.9 - Estações de trabalho	261
5.8.9.1 - Tipos de estações de trabalho	262
5.8.9.2 - Seleção de uma estação de trabalho	264
5.8.9.3 - Transformações de visualização (coordenadas NDC para DC)	266
5.8.9.4 - Segmentos	268
5.8.9.5 - Dispositivos de entrada	269
5.8.9.6 - WISS - Armazenador de segmentos independente de estação de trabalho	270
5.8.9.7 - Metarquivos	272
5.9 - GKS-3D Uma extensão tridimensional do GKS	274
5.9.1 - Introdução	274
5.9.2 - Estrutura conceitual	275
5.9.3 - Primitivas de saída	276
5.9.4 - Atributos	278

5.9.5 - Transformações e visualização	280
5.9.5.1 - Transformação de normalização	281
5.9.5.2 - Transformações de segmento e inserção	282
5.9.5.3 - Transformação de visualização	282
5.9.5.4 - Transformação de estação de trabalho	285
5.9.6 - Segmentação	285
5.9.7 - Entrada	286
5.9.8 - Funções de consulta	287
5.10 - PHIGS - "Programmer's Hierarchical Interactive Graphics System"	287
5.10.1 - Introdução	287
5.10.2 - A organização do modelo	289
5.10.3 - Conteúdo do modelo	290
5.10.4 - Modificação do modelo	290
5.10.5 -A relação de PHIGS com GKS	292
5.10.6 - Areas funcionais de PHIGS	296
5.10.6.1 - Hierarquia de estruturas	296
5.10.6.2 - Associação de atributos em tempo de per- curso	296

5.10.6.3 - Nomes de conjuntos e filtros	299
5.10.6.4 - Processo de transformação	300
5.10.6.5 - Edição de estruturas	303
5.11 - Conclusão do estudo sobre padronização gráfica	304
<u>CAPÍTULO 6 - AVALIAÇÃO DE RESULTADOS E OBJETIVOS</u> <u>ALCANÇADOS</u>	309
6.1 - Introdução	309
6.2 - Etapas de desenvolvimento	309
6.3 - Fluxograma do modelador geométrico	314
6.3.1 - Fluxograma "menu principal"	314
6.3.2 - Fluxograma "sub-rotina inicializa parte"	316
6.3.3 - Fluxograma "sub-rotina operações com sólidos"	319
6.3.4 - Fluxograma "sub-rotina sólido combinado"	320
6.3.5 - Fluxograma "sub-rotina sólido transformado" ..	323
6.3.6 - Fluxograma "sub-rotina sólido primitivo"	327
6.3.7 - Fluxograma "sub-rotina varredura"	328
6.3.8 - Fluxograma "sub-rotina varredura rotacional" .	330

6.3.9 - Fluxograma "sub-rotina varredura translacional"	332
6.3.10 - Fluxograma "sub-rotina definir contorno" ...	335
6.3.11 - Fluxograma "sub-rotina manipulação de entidades 2D"	337
6.3.12 - Fluxograma "sub-rotina sombreadimento"	339
6.3.13 - Fluxograma "sub-rotina definir parâmetros" .	342
6.3.14 - Fluxograma "sub-rotina banco de dados"	345
<u>CAPÍTULO 7 - CONCLUSÕES E RECOMENDAÇÕES</u>	349
REFERÊNCIAS BIBLIOGRÁFICAS	357
APÊNDICE A - BIBLIOGRAFIA COMPLEMENTAR	

LISTA DE FIGURAS

	<u>Pág.</u>
1.1 - Ambiente do modelador geométrico dentro do sistema CAD utilizado	2
2.1 - Imagem simplificada de um SBD	5
2.2 - Exemplo de dados operacionais	7
2.3 - Campo	11
2.4 - Registro	12
2.5 - Arquivo	13
2.6 - Arquitetura de um SBD	14
2.7 - Visão Relacional	18
2.8 - Sólidos compostos do exemplo	19
2.9 - Produto Cartesiano	21
2.10 - Relação COMBINAÇÃO	22
2.11 - Visão hierarquica da relação PARTICIPAÇÃO	23
2.12 - Arquitetura hierárquica	25
2.13 - Arquitetura de um sistema IMS	26
2.14 - Abordagem em Rede	28

2.15 - Arquitetura de um sistema DBTG	30
2.16 - Modelo da estrutura de armazenamento do modelador geométrico	41
2.17 - Representação de um sólido no banco de dados implantado	52
3.1 - Uma curva paramétrica	58
3.2 - Elementos vetoriais de uma curva paramétrica ...	60
3.3 - Reta pc	65
3.4 - Curva Spline	66
3.5 - Curvas de Bézier	68
3.6 - Curva cúbica de Bézier	70
3.7 - Funções de blending para $n=3$	71
3.8 - Funções de blending B-Spline para $n=5$, $k=3$	73
3.9 - Curva B-Spline não periódica: $n=5$, $k=3$	75
3.10 - Curva B-Spline periódica com 4 pontos de controle	76
3.11 - Retalho paramétrico de superfície	78
3.12 - Coordenadas paramétricas e coordenadas xy de um plano	80
3.13 - Esfera paramétrica	81

3.14 - Superfície paramétrica de revolução	82
3.15 - Superfície bicúbica mapeada no espaço objeto a partir de seus componentes no espaço paramétrico	85
3.16 - Nomenclatura para uma superfície bicúbica	86
3.17 - Espaço paramétrico de uma superfície	88
3.18 - Equação vetorial de um plano	90
3.19 - Superfície cilíndrica	91
3.20 - Superfície reguada	93
3.21 - Superfície de revolução	94
3.22 - Análise de uma superfície de revolução	95
3.23 - Superfície esférica	97
3.24 - Superfície cúbica de Bézier	99
3.25 - Superfícies B-Spline	102
3.26 - Elementos de contorno de um sólido paramétrico	104
3.27 - Sólido retangular paramétrico	106
3.28 - Coeficientes geométricos do cubo unitário	109
3.29 - Vetores tangentes do sólido paramétrico	110
3.30 - Espaço paramétrico de um sólido paramétrico ...	112

3.31 - Condições de continuidade	113
3.32 - Convenção do sentido de rotação positivo	118
3.33 - Distância entre o observador e a origem	124
3.34 - Ângulo de Azimuth	125
3.35 - Ângulo de incidência	125
3.36 - Ângulo de torsão	126
3.37 - Visualização de cena	127
3.38 - Volume de visualização em perspectiva	130
3.39 - Volume de visualização ortográfico	132
3.40 - Sistema geométrico	134
3.41 - (a) Conjunto não regular; (b) Conjunto não semi-analítico	139
3.42 - Modelo procedural simples	143
3.43 - Árvore binária para $D=(A \cup B)-C$	144
3.44 - Modelo booleano de uma peça mecânica	145
3.45 - Decomposição celular	146
3.46 - Exemplos de representação por varredura	148
3.47 - Representações por varredura dimensionalmente não homogêneas	149

3.48 - Características de uma curva PD	150
3.49 - Representação CSG	153
3.50 - Modelos em arame de 2 1/2 dimensão não ambíguos	157
3.51 - Ambigüidade do modelo em arame	157
4.1 - Componentes de um sistema CAD/CAM	160
4.2 - Software do sistema	164
4.3 - Espaços dos modos Model e Draw	166
4.4 - Modelos e folhas de desenho	167
4.5 - Sistema de coordenadas Model e Draw	168
4.6 - Modelos, vistas e drawings	171
5.1 - O ambiente da computação gráfica	185
5.2 - Ambiente da computação gráfica em um nível mais baixo	186
5.3 - Modelo de referência da computação gráfica e pa- drões gráficos	187
5.4 - Interface Independente/Dependente de dispositivo DI/DD	193
5.5 - Estrutura de saída da parte CORE independente de dispositivo	196

5.6 - Estrutura de entrada da parte CORE independente de dispositivo	200
5.7 - A relação de CGM com um ambiente de aplicações gráficas	207
5.8 - Estrutura de CGM	213
5.9 - CGI dentro de um sistema gráfico	219
5.10 - Indexação da tabela de cores	243
5.11 - Primitiva "cell array"	244
5.12 - Preenchimento de áreas por "fill area" usando o método de paridade	246
5.13 - A transformação de normalização	250
5.14 - Sequência de transformações nos segmentos antes da visualização	254
5.15 - Transformação de coordenadas NDC para coordenadas do mundo WCS	256
5.16 - A janela e a vista da transformação de visualização de uma estação de trabalho	267
5.17 - Sequência de transformações geométricas de coordenadas WCS a DC	268
5.18 - O processo de transformação do GKS-3D	280
5.19 - Volumes de visualização e tipos de projeção ...	283

5.20 - Um típico sistema de aplicações com gráficos ..	289
5.21 - O processo de transformação PHIGS	291
5.22 - Comparação da topologia de dados entre GKS e PHIGS	293
5.23 - Comparação da associação de atributos em GKS e PHIGS	295
5.24 - A organização de dados hierárquica de PHIGS ...	296
5.25 - O modelador geométrico dentro do modelo de re- ferência da padronização gráfica	304
6.1 - Fluxograma "menu principal"	315
6.2 - Fluxograma "sub-rotina inicializa parte"	317
6.3 - Fluxograma "sub-rotina operações com sólidos" ..	319
6.4 - Fluxograma "sub-rotina sólido combinado"	321
6.5 - Continuação do fluxograma "sub-rotina sólido combinado"	322
6.6 - Fluxograma "sub-rotina sólido transformado"	324
6.7 - Continuação do fluxograma "sub-rotina sólido transformado"	325
6.8 - Fluxograma "sub-rotina sólido primitivo"	327
6.9 - Fluxograma "sub-rotina varredura"	329

6.10 - Fluxograma "sub-rotina varredura rotacional" ..	331
6.11 - Fluxograma "sub-rotina varredura translacional"	333
6.12 - Fluxograma "sub-rotina definir contorno"	335
6.13 - Fluxograma "sub-rotina manipulação de entidades 2D"	338
6.14 - Fluxograma "sub-rotina sombreamento"	340
6.15 - Fluxograma "sub-rotina definir parâmetros"	343
6.16 - Fluxograma "sub-rotina banco de dados"	346

CAPÍTULO 1

INTRODUÇÃO

O propósito do presente trabalho é o desenvolvimento de um modelador geométrico para a construção de sólidos complexos, compostos a partir de sólidos mais simples gerados por varredura em uma estação de trabalho CAD (Projeto Auxiliado por Computador).

Os sólidos são construídos seguindo uma estrutura hierárquica em forma de árvore binária, cujas folhas ou elementos do nível mais inferior são figuras bidimensionais (ou primitivas gráficas básicas), usados para a geração de sólidos através da técnica conhecida como varredura ("sweeping") rotacional ou translacional em torno de um eixo de rotação ou ao longo de uma direção, respectivamente.

Os sólidos gerados por varredura são então compostos entre si para a obtenção da forma final desejada.

O modelador geométrico implantado, é um programa de aplicação que implementa uma interface interativa entre o usuário de uma estação de trabalho CAD e um pacote gráfico (biblioteca gráfica) próprio de tal sistema CAD.

A figura -(Fig. 1.1) abaixo, descreve o ambiente do modelador geométrico implantado dentro do sistema CAD utilizado.

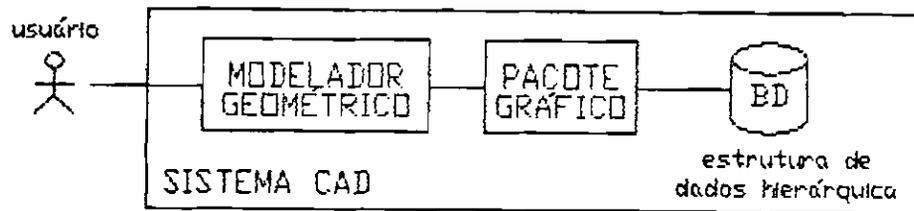


Fig. 1.1 - Ambiente do modelador geométrico dentro do sistema CAD utilizado.

O modelador geométrico, implementa uma estrutura de dados hierárquica que descreve o processo de construção dos sólidos. Esta estrutura hierárquica é armazenada e recuperada de um banco de dados gráfico.

Neste contexto, o banco de dados manipulado, se refere apenas ao local de armazenamento dos arquivos da estrutura de dados implementada e não representa um Sistema de Banco de Dados SBD. Deve se ressaltar que dentro do sistema CAD utilizado, não existe suporte ou relação com algum SBD.

O sistema CAD utilizado, foi o sistema CAD/CAM Computervision's Designer V-X, o pacote gráfico é o CADDs 4X, e o modelador geométrico foi implementado na linguagem VARPRO 2 própria do sistema.

Inicialmente, com o objetivo de definir um modelo de estrutura de dados apropriado, é apresentado um estudo teórico sobre banco de dados, conceitos básicos e tipos de abordagens de estruturas de dados, concluindo com a escolha da abordagem mais conveniente ao modelo de estrutura de dados necessária e apresentando a estrutura de dados usada pelo modelador geométrico aplicável ao modelamento de sólidos dentro de um sistema CAD.

Após o estudo do banco de dados, é introduzida a base teórica do modelamento geométrico, abordando curvas, superfícies, sólidos e transformações, assim como os fundamentos do modelamento de sólidos e os principais esquemas de representação de sólidos.

Com o objetivo de estender a implementação do modelador geométrico desenvolvido, usando outros pacotes gráficos padronizados, segue um estudo teórico sobre Padronização Gráfica, concluindo com uma análise da viabilidade desta transportabilidade.

Finalmente, discute-se sobre a implantação do trabalho no sistema CAD utilizado, o software gráfico desenvolvido para a implementação do modelador geométrico e a sua estrutura de dados associada, e os resultados atingidos.

CAPÍTULO 2

BANCO DE DADOS

2.1 - CONCEITOS BÁSICOS

2.1.1 - SISTEMA DE BANCO DE DADOS

Um Sistema de Banco de Dados SBD, é um sistema capaz de armazenar e manipular dados, atendendo a vários usuários, os quais têm acesso aos dados através de programas de aplicação.

A figura abaixo (Fig. 2.1), ilustra uma imagem simplificada de um SBD:

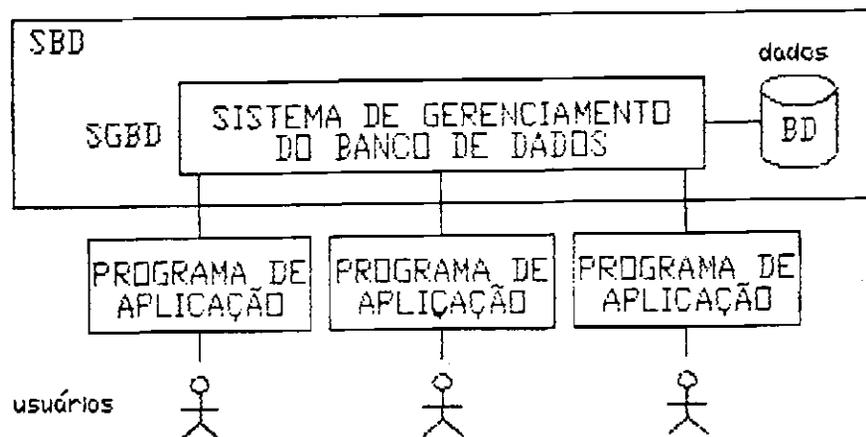


Fig. 2.1 - Imagem simplificada de um SBD.

O SBD envolve:

1. Dados
2. Hardware
3. Software
4. Usuários

2.1.1.1 - DADOS

Os dados são armazenados por um ou mais banco de dados (BD). Assume-se a existência de apenas um BD o qual contém todos os dados.

O BD é tanto integrado como compartilhado.

Por integrado, entende-se como a unificação de diversos arquivos de tal maneira que se elimine total ou parcialmente qualquer redundância de dados entre estes arquivos.

Por compartilhado, entende-se que os mesmos dados podem ser compartilhados entre diversos usuários, o que significa que cada um destes usuários pode ter acesso aos mesmos dados e usá-los para finalidades diferentes.

Se este acesso ocorre ao mesmo tempo, então o compartilhamento é chamado concorrente.

Um BD é uma coleção de dados operacionais, os quais são diferentes dos dados de entrada, dados de saída e outros tipos de dados.

Os dados operacionais representam associações e relações entre as entidades do BD, e não incluem dados de entrada nem dados de saída.

A figura (Fig. 2.2) abaixo, é um exemplo de dados operacionais de um sistema gráfico de modelamento de sólidos:

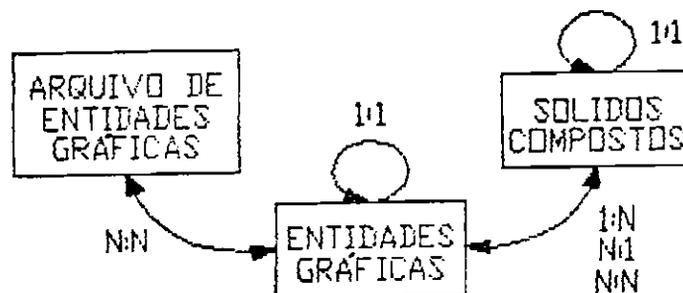


Fig. 2.2 - Exemplo de dados operacionais.

Na figura (Fig. 2.2) acima, temos:

- entidades gráficas: cilindro, cubo, cone, esfera ou qualquer sólido, todos gerados por varredura. Por exemplo, cone gerado pela revolução de uma reta em torno de um eixo ou cubo gerado pela varredura translacional de um quadrado ao longo de uma direção;
- sólidos compostos, a partir das entidades gráficas usando composição booleana entre sólidos: união, diferença e intersecção, ou a partir de sólidos compostos mais simples. Por exemplo, um colar gerado pela união de várias esferas;

- arquivos de entidades gráficas, local onde as entidades gráficas são armazenadas;
- entidades representadas por retângulos. Uma entidade é a unidade básica do BD. As setas representam associações e relacionamentos interligando as entidades e portanto constituem os dados operacionais.

Como tipos de relacionamentos, observando a figura (Fig. 2.2) anterior, podem ser citados:

1. Relacionamentos 1:N

Entidades gráficas - sólidos compostos. Uma entidade faz parte de vários sólidos e estes compartilham uma mesma entidade.

2. Relacionamentos N:1

Entidades gráficas - sólidos compostos. Várias entidades compõem um sólido, e este é composto por várias entidades.

3. Relacionamentos 1:1

Sólidos compostos - sólidos compostos. Um sólido faz parte da composição de outro sólido.

4. Relacionamentos bidirecionais N:N

Arquivos de entidades - entidades gráficas. Cada arquivo fornece entidades diferentes e cada entidade é fornecida por arquivos diferentes.

Os relacionamentos e as entidades são parte dos dados operacionais.

2.1.1.2 - HARDWARE

O hardware do SBD representa a memória física (disco, unidade de fita magnética) nos quais reside o BD.

2.1.1.3 - SOFTWARE

É a interface entre o BD físico e os usuários do sistema. Conhecido como Sistema de Gerenciamento de Banco de Dados SGBD, e é implementado em uma linguagem de alto nível.

2.1.1.4 - USUÁRIOS

Podem ser de 3 tipos:

1. Programador de Aplicações

Escreve programas de aplicação que utilizam o BD, criando, recuperando ou alterando informações.

2. Usuário Final

O qual tem acesso ao BD a partir de um terminal usando uma linguagem de consulta de alto nível.

3. Administrador do Banco de Dados ABD

Pessoa ou grupo do controle global do BD.

2.1.2 - A NECESSIDADE DO BANCO DE DADOS

Um SBD proporciona o controle centralizado dos dados operacionais do sistema (por exemplo, uma empresa ou fábrica) ao qual pertence. O ABD é responsável pelo controle central dos dados operacionais.

Entre as vantagens do controle centralizado dos dados, podem ser citadas:

- Reduz a redundância dos dados armazenados.
- Evita a inconsistência de dados: o ingresso de informação no sistema por duas entradas distintas impossibilitando sua alteração nas duas estruturas.
- Os dados podem ser compartilhados: os mesmos dados podem operar com aplicações diferentes.
- Podem ser aplicadas restrições de segurança, garantindo vias únicas de acesso e diferentes verificações para cada tipo de usuário.
- A integridade pode ser mantida, garantindo dados precisos, evitando inconsistência e redundância.

- As necessidades conflitantes na procura dos dados no BD podem ser balanceadas, dando acesso mais rápido às aplicações mais importantes e mais lento para as outras aplicações.

2.1.3 - ESTRUTURA DE ARMAZENAMENTO

A estrutura de armazenamento de um SBD, ao qual somente o ABD tem acesso, é composta por:

1. Campo

Menor unidade de armazenamento de dados que possui um nome e um determinado tipo de dado. Por exemplo, o campo com nome E# (número de identificação da entidade gráfica) e tipo de dado numérico, como mostra a figura (Fig. 2.3) abaixo:

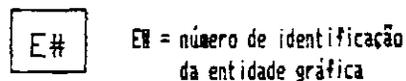


Fig. 2.3 - Campo.

2. Registro

Composto por vários campos e tem um nome associado. Por exemplo, os registros com nome ENTIDADE (entidade gráfica) e SOLIDO (sólido composto), que são, por sua vez, compostos por vários campos como mostra a figura (Fig. 2.4) a seguir:

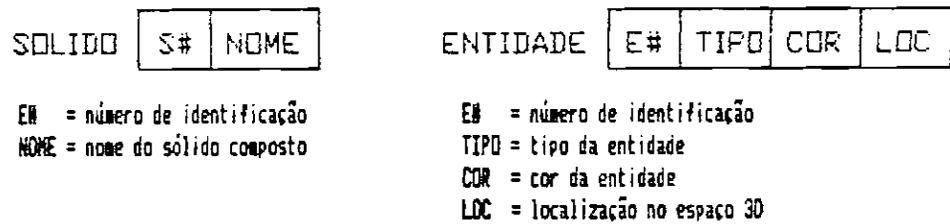


Fig. 2.4 - Registro.

Cada entidade gráfica tem um registro diferente e o ABD pode alterar os campos do registro de uma determinada entidade gráfica.

Podem ser concatenados dois registros em um só, por exemplo, os registros (E#,TIPO,COR,LOC) e (S#,NOME) para obter o registro (E#,TIPO,COR,LOC,S#,NOME).

Ou, se pode dividir um registro em dois registros menores: o registro (E#,TIPO,COR,LOC) pode ser dividido nos registros (E#,TIPO,COR) e (E#,TIPO).

Assim, as partes menos utilizadas podem ser armazenadas em dispositivos mais lentos.

3. Arquivo

Composto de vários registros e tem associado um nome. Também é conhecido como Registro Lógico. Por exemplo, os arquivos de nome ARQ_ENT (arquivo de entidades gráficas) e ARQ_SOL (arquivo de sólidos compostos) como mostrados na figura (Fig. 2.5) a seguir, são compostos por vários registros:

ARQ_ENT	E#	TIPO	COR	LOC	ARQ_SDL	S#	NOME
	1	RETA	VERDE	(0,0,0)		1	CADEIRA
	2	ARCO	AZUL	(10,20,30)		2	MESA
	3	CILINDRO	BRANCO	(-10,30,40)		3	CARRO

Fig. 2.5 - Arquivo.

O arquivo pode ter registros de diversos campos e ser armazenado de diferentes maneiras. Pode estar organizado sequencialmente ou não, e o acesso pode ser, ou não, via endereço randômico.

2.1.4 - INDEPENDÊNCIA DE DADOS

Um sistema dependente dos dados é aquele onde é impossível mudar a estrutura interna de armazenamento dos dados, ou a estratégia de acesso aos dados, sem afetar a aplicação.

Um SBD não pode ser dependente dos dados em virtude de:

- Aplicações diferentes podem precisar de visões diferentes do mesmo dado.
- O ABD tem que ter liberdade para modificar a estrutura de armazenamento ou a estratégia de acesso em resposta a necessidades de mudanças, sem ter que mudar as aplicações existentes.

A independência de dados, permite que o ABD possa alterar a estrutura de armazenamento do BD sem afetar as aplicações existentes. Alterando a estrutura do BD, o BD pode crescer ou diminuir dependendo da aplicação.

2.1.5 - ARQUITETURA DE UM SBD

A figura (Fig. 2.6) abaixo, mostra a arquitetura do SBD:

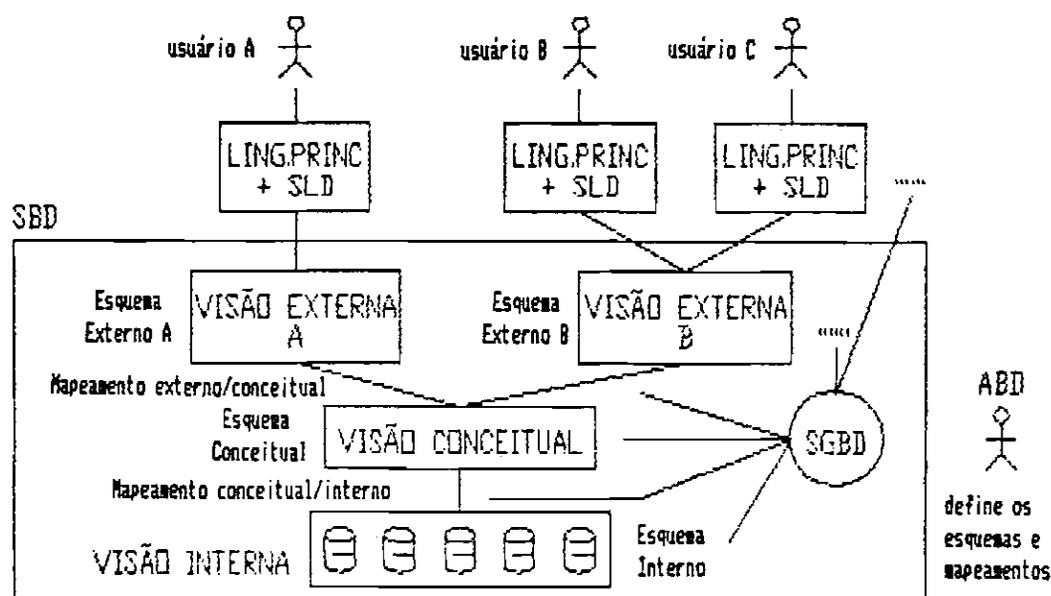


Fig. 2.6 - Arquitetura de um SBD.

Como mostra a figura (Fig. 2.6) acima, o SBD é dividido em 3 níveis gerais:

1. Nível interno

Onde os dados estão armazenados fisicamente.

2. Nível externo

Visão externa dos dados por cada usuário.

3. Nível conceitual

Nível de interface entre os outros dois níveis. É único e atende a várias visões externas.

Os usuários são programadores de aplicações, programando em uma linguagem principal de alto nível a qual inclui uma sublinguagem de dados SLD direcionada para os objetos e operações no BD.

O sistema pode suportar várias linguagens principais e várias SLD. A SLD compõe-se de duas linguagens: linguagem de definição de dados LDD voltada para a descrição dos objetos do BD, e a linguagem de manipulação de dados LMD.

A cada nível está associada uma visão, ou maneira de visualizar o BD, como segue :

1. Visão externa

Visão do usuário, definida por meio de um esquema externo o qual é escrito em LDD externa.

2. Visão conceitual

Representação do conteúdo completo do BD. É uma visão de como os dados realmente são e não como os usuários os vêem pelas restrições da linguagem. Definida por meio de um esquema conceitual escrito em LDD conceitual.

3. Visão interna

É uma representação de baixo nível de todo o BD, porém acima do nível físico, pois não chega a lidar com hardware. É descrita por meio de um esquema interno escrito em LDD interna.

O mapeamento conceitual/interno define a correspondência entre a visão conceitual e o BD armazenado. O mapeamento externo/conceitual define a correspondência entre uma determinada visão externa e a visão conceitual.

O Sistema de Gerenciamento do Banco de Dados SGBD é o software que manipula todos os acessos ao BD. Conceitualmente ocorre o seguinte:

1. O usuário solicita o BD usando alguma LMD
2. O SGBD intercepta a solicitação e a interpreta
3. O SGBD consulta o esquema externo, o mapeamento externo/conceitual e o esquema interno
4. O SGBD executa as operações necessárias no BD armazenado

As responsabilidades do ABD são:

- Decidir quais dados devem ser mantidos no BD, definindo o conteúdo do BD e escrevendo o esquema conceitual na LDD conceitual. Este esquema é compilado e usado pelo SGBD.
- Decidir a estrutura de armazenamento e a estratégia de acesso, escrevendo a estrutura de dados em LDD interna e o mapeamento estrutura de dados/esquema conceitual usando LDD interna e LDD conceitual. Define estratégias de backup e recuperação de arquivos.
- Ser a interface com os usuários, especificando para cada usuário o mapeamento externo/conceitual usando várias LDD externas. A interface com o usuário é a fronteira do sistema além da qual tudo se torna invisível ao usuário.

2.2 - ESTRUTURAS DE DADOS DE UM SBD

2.2.1 - INTRODUÇÃO

O acesso do usuário ao BD através de uma LMD, implica na definição de uma estrutura de dados a ser usada.

Existem 3 tipos de abordagens diferentes para a definição e manipulação de uma estrutura de dados:

1. Abordagem Relacional
2. Abordagem Hierárquica
3. Abordagem em Rede

2.2.2 - ABORDAGEM RELACIONAL

Seja como exemplo, um BD contendo informações sobre sólidos compostos, entidades gráficas e participação das entidades gráficas na composição dos sólidos. Uma visão relacional neste exemplo, representaria cada informação em uma tabela ou relação.

A figura (Fig. 2.7) abaixo, mostra 3 tabelas ou relações: SOLIDOS, ENTIDADES e PARTICIPAÇÃO, descrevendo a estrutura de dados necessária para atender os seguintes requisitos:

SOLIDOS						PARTICIPAÇÃO		
S#	NOME	E#	TIPO	COR	LOC	S#	E#	QUANT
S1	MESA	E1	CILINDRO	AMARELO	(0,0,0)	S1	E1	4
S2	CONECTOR	E2	CUBO	VERMELHO	(11,20,35)	S1	E2	16
S3	COLAR	E3	ESFERA	VERDE	(-15,35,80)	S1	E3	1
		E4	COPE	BRANCO	(50,50,-50)	S2	E1	3
						S2	E2	1
						S3	E2	2
						S3	E3	8

Fig. 2.7 - Visão Relacional.

- cada sólido possua um número #S e um nome NOME
- cada entidade possua um número #E, um tipo de entidade TIPO, uma cor COR e uma posição no espaço LOC
- seja desejado obter informação da participação de uma determinada combinação SOLIDOS/ENTIDADES

Uma relação é definida como uma coleção de conjuntos D_1, D_2, \dots, D_n , onde R é uma relação nos n conjuntos se R for um conjunto de n -túplas ordenadas (d_1, d_2, \dots, d_n) tais que $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$. Os conjuntos D_1, D_2, \dots, D_n , são os domínios de R . O valor de n é o grau de R .

A relação PARTICIPAÇÃO, como mostra a figura (Fig. 2.8) abaixo, descreve:

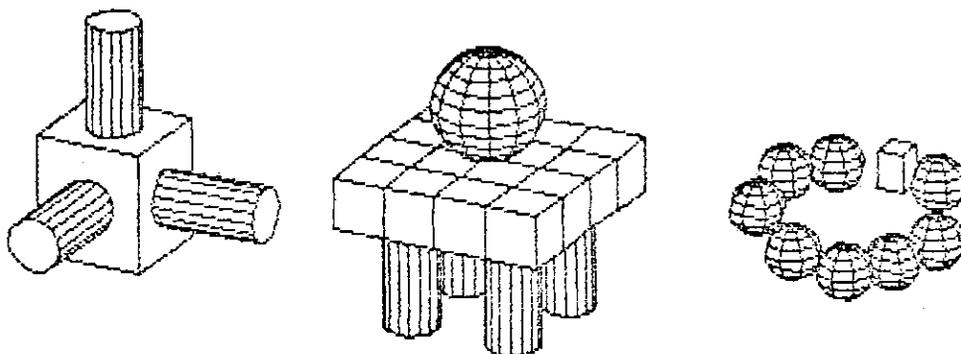


Fig. 2.8 - Sólidos compostos do exemplo.

- 1 mesa composta por 4 cilindros (pés), 16 cubos (que unidos definem a tábua) e 1 esfera (sobre a mesa)

- 1 conector composto por 3 cilindros e 1 cubo
- 1 colar composto por 8 esferas e 1 cubo (o fecho)

Na figura (Fig. 2.7) anterior temos 3 relações. A relação ENTIDADES tem grau 4 e a relação PARTICIPAÇÃO tem grau 3.

Assim mesmo, a relação ENTIDADES tem 4 domínios: E#, TIPO, COR e LOC. O domínio COR é o conjunto de todas as cores.

Cada linha da tabela representa uma n-tupla (tupla de n domínios) da relação. A cardinalidade da relação é o número de tuplas, na relação PARTICIPAÇÃO a cardinalidade é 7.

O produto cartesiano de uma coleção de conjuntos D_1, D_2, \dots, D_n , escrito como $D_1 \times D_2 \times \dots \times D_n$, é o conjunto de todas as possíveis n-tuplas ordenadas (d_1, d_2, \dots, d_n) tal que $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$.

A figura (Fig. 2.9) a seguir, mostra o produto cartesiano dos conjuntos S# e E#:

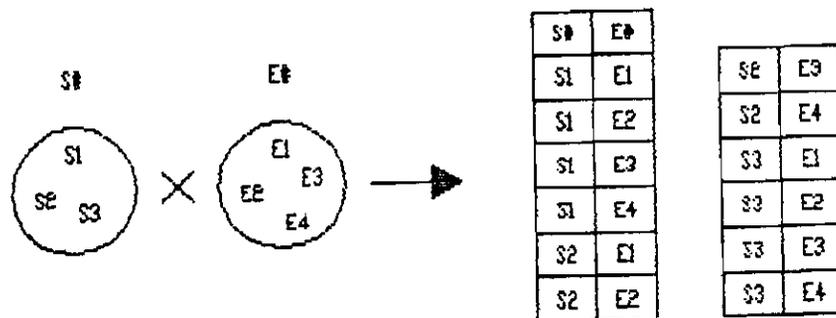


Fig. 2.9 - Produto Cartesiano.

Logo R é uma relação dos conjuntos D_1, D_2, \dots, D_n se ela for um subconjunto do produto cartesiano $D_1 \times D_2 \times \dots \times D_n$.

As tóplas de uma relação não são ordenadas, pois uma relação é um conjunto e conjuntos não são ordenados, porém o usuário de um BD pode ordená-las.

Em contraste, os domínios de uma relação são ordenados, pois o j -ésimo elemento de cada n -túpla é retirado da j -ésima coluna.

Existe a diferença entre um domínio e colunas (ou atributos) que são retiradas daquele domínio. Um atributo usa elementos de um domínio dentro de uma relação.

A figura (Fig. 2.10) a seguir, mostra a relação COMBINAÇÃO com 3 atributos ENTIDADE1, ENTIDADE2 E OPERAÇÃO, e apenas 2 domínios $E\#$ e OPERAÇÃO, pois os elementos dos atributos (colunas) ENTIDADE1 e ENTIDADE2 pertencem a um mesmo domínio $E\#$.

COMBINAÇÃO	ENTIDADE 1	ENTIDADE 2	OPERAÇÃO
	E1	E1	UNIÃO
	E1	E2	INTERSECÇÃO
	E3	E3	DIFERENÇA

Fig. 2.10 - Relação COMBINAÇÃO.

Define-se "chave" de uma relação a um dos atributos cujos valores são únicos dentro da relação e portanto podem ser usados para identificar as tóplas daquela relação.

Assim, na relação ENTIDADES pode-se usar como chave o atributo E#, pois cada tópla de ENTIDADES tem um valor distinto de E#.

A Abordagem Relacional considera arquivos, com certas limitações, como relações matemáticas, capazes de representar de uma maneira uniforme todas as informações em tabelas.

O relacionamento entre as tabelas é abordada pela Álgebra Relacional a qual é um conjunto de operações e relações onde cada operação usa uma ou mais relações como seus operandos e produz outra relação como seu resultado.

Os operadores tradicionais da Álgebra Relacional, União, Intersecção, Diferença e Produto Cartesiano, operam sobre relações ao invés de conjuntos.

Outros operadores relacionais são a Seleção, Projeção, Junção e Divisão.

Entre os sistemas relacionais mais conhecidos podem ser citados: MAGNUM da Tymshare, QUERY BY EXAMPLE da IBM, NOMAD da NCSS, SYSTEM R da IBM Research e INGRES.

2.2.3 - ABORDAGEM HIERÁRQUICA

A figura abaixo (Fig. 2.11) mostra a visão hierárquica do exemplo da seção (2.8) anterior. Representa a relação PARTICIPAÇÃO com ENTIDADES acima de SÓLIDOS:

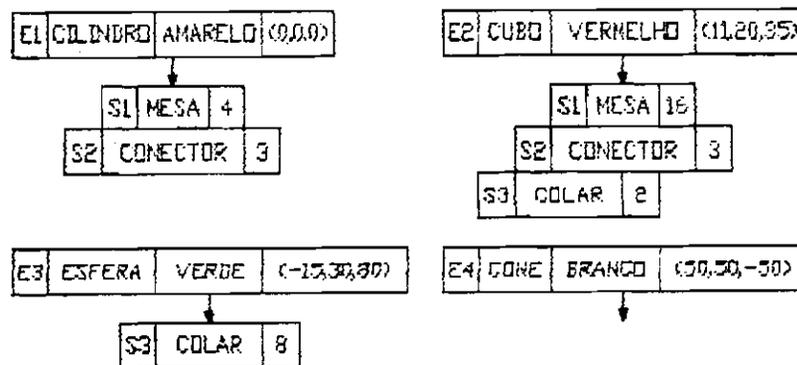


Fig. 2.11 - Visão hierárquica da relação PARTICIPAÇÃO.

O usuário vê 4 árvores separadas, uma para cada entidade. Cada árvore consiste de um registro de entidades juntamente com um conjunto de registros de sólidos subordinados.

Cada registro sólido inclui mais um campo, a quantidade de participação. Assim, a árvore do registro da entidade E1 expressa que E1 participa no sólido S1 uma vez e no sólido S2 tres vezes.

O conjunto de sólidos para uma determinada entidade pode conter qualquer quantidade de membros inclusive 0, como para a entidade E4.

No caso, entidade é a raiz que pode conter qualquer número de dependentes, cada um dos quais pode conter qualquer quantidade de dependentes e assim por diante.

Cada entidade E_i está relacionada com o sólido S_i e com a participação correspondente que liga E_i com S_i na relação PARTICIPAÇÃO.

A visão relacional foi assemelhada a 3 arquivos simples, porém a visão hierárquica consiste de um só arquivo contendo registros organizados em 3 árvores separadas.

A LMD hierárquica contém ponteiros conectando ocorrências entre registros, e pode ser considerada como uma coleção de operações sobre estas árvores.

As consultas para cada tipo de entidade são diferentes, ou assimétricas, por se ter árvores diferentes. A assimetria é um grande obstáculo na abordagem hierárquica, tanto maior quanto mais complexa é a árvore.

As hierarquias são úteis na modelagem de estruturas hierárquicas onde os relacionamentos sejam do tipo 1:N.

A arquitetura hierárquica como mostra a figura (Fig. 2.12) a seguir, é a representação de uma coleção de banco de dados físicos BDF:

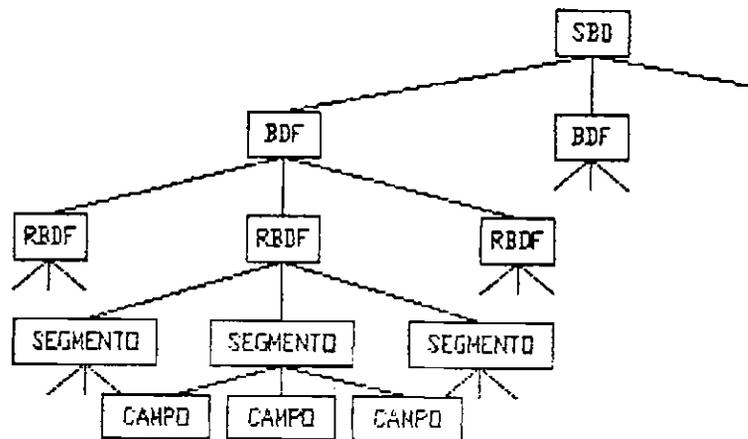


Fig. 2.12 - Arquitetura hierárquica.

Cada BDF por sua vez é uma coleção de registros de banco de dados físicos RBDF hierárquicos. Cada RBDF é um arranjo hierárquico de segmentos cada um dos quais é uma coleção de campos.

Assim, para o exemplo da figura (Fig. 2.11) anterior, o BDF seria todo o banco de dados exemplo envolvendo as 4 árvores hierárquicas que descrevem as relações SÓLIDOS, ENTIDADES e PARTICIPAÇÃO.

Cada uma das 4 árvores hierárquicas seria um RBDF, cada um dos quais é uma coleção de segmentos registros de uma determinada relação e estes registros por sua vez estão divididos em campos. No caso o segmento entidade está dividido nos campos E#, TIPO, COR e LOC.

Como exemplo de uma arquitetura hierárquica apresenta-se a arquitetura do sistema IMS Information Management System da IBM, na figura (Fig. 2.13) a seguir:

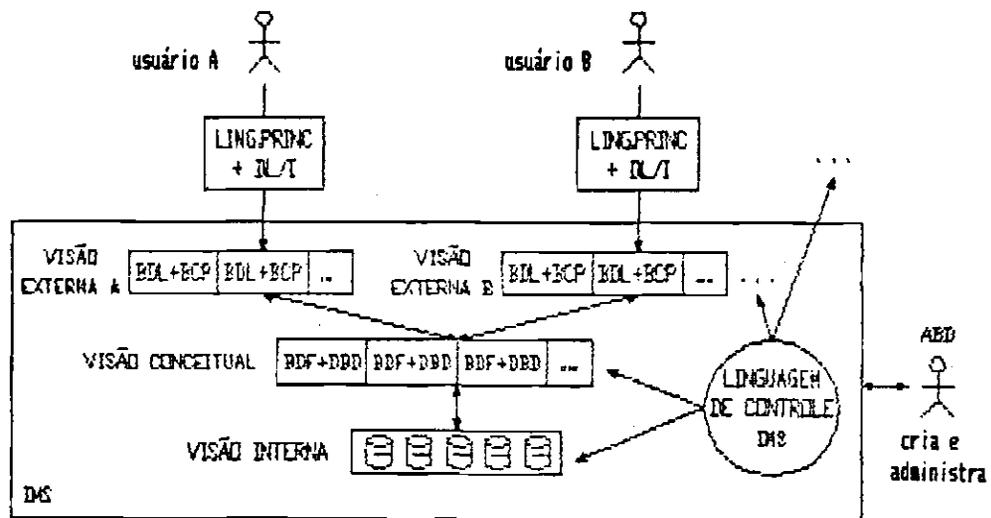


Fig. 2.13 - Arquitetura de um sistema IMS.

Os dados armazenados consistem de diversos bancos de dados, não apenas um. Esta é a visão conceitual.

Cada BDF é definido por uma descrição de banco de dados DBD. O mapeamento entre o BDF e o armazenamento é também especificado na DBD.

Portanto o conjunto de todas as DBD corresponde ao esquema conceitual mais (parte da) definição associada do mapeamento conceitual/interno.

A visão externa do usuário é uma coleção de bancos de dados lógicos BDL, na qual cada BDL é um subconjunto do BDF correspondente.

Cada BDL é definido juntamente com seu mapeamento ao BDF por meio do bloco de comunicação com o programa BCP.

Todos os BCP de um usuário correspondendo ao esquema externo mais a definição de mapeamento associada, é um bloco de especificação do programa BEP.

Logo, um BDL é uma visão particular do usuário à uma hierarquia do RBDF. O BDL do usuário poderia ser apenas uma das 4 árvores, onde cada segmento da árvore escolhida é definido como um registro de banco de dados lógico RBDL.

Os usuários usam uma linguagem principal (PL/1, COBOL, etc.) através da qual pode ser chamada a linguagem DL/I (Data Language I) de manipulação de dados do IMS.

Entre os sistemas hierárquicos mais conhecidos, podem ser citados: o IMS da IBM Information Managment System, o MARK IV da Informatics, o SYSTEM 2000 da MRI, e o TDMS da SDC Time-Shared Data Managment System.

2.2.4 - ABORDAGEM EM REDE

A seguir, a figura (Fig. 2.14), mostra uma visão em rede do banco de dados do exemplo em uso (SÓLIDOS, ENTIDADES E PARTICIPAÇÃO):

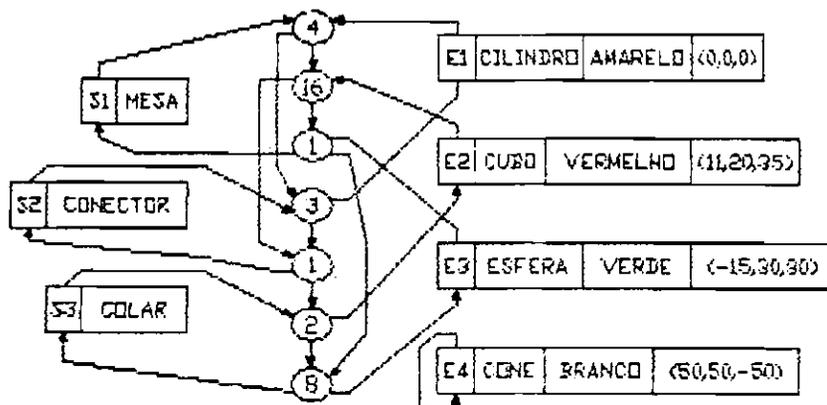


Fig. 2.14 - Abordagem em Rede.

Assim como na abordagem hierárquica os dados estão representados por registros e ponteiros.

Porém, a abordagem em rede pode ter qualquer quantidade de superiores imediatos (um filho pode ter vários pais, em contraste com a abordagem hierárquica onde um filho só pode ter 1 pai). Por isto, é possível implementar relações N:N mais diretamente do que a abordagem hierárquica.

Além dos tipos de registro que representam sólidos e entidades é introduzido um novo tipo de registro chamado "conector".

O conector representa a associação PARTICIPAÇÃO entre um sólido e uma entidade contendo dados que descrevem esta relação (quantas entidades participam na composição do sólido).

Todas as ocorrências de conector de um determinado sólido (ou entidade) estão colocadas em uma cadeia que começa e termina naquele sólido (ou na entidade). Cada conector encontra-se em duas cadeias, uma de sólidos e a outra de entidades.

Por exemplo, na figura (Fig. 2.14) anterior, vemos que a cadeia, que começa no sólido S1 passa pelos conectores 4,16 e 1 e retorna ao sólido S1, é composta por $4+16+1$ entidades, sendo 4 de E1, 16 de E2 e 1 de E3, observe que estes conectores são visitados por estas entidades.

Assim mesmo, a cadeia que começa em S2 passa pelos conectores 3 e 1 e volta para S2, representa que S2 é composto de $3+1$ entidades sendo 3 de E1 e 1 de E2.

A cadeia que parte de E1, passa pelos conectores 4 e 3 e volta a E1, encontrando-se com S1 no conector 4 e com S2 no conector 3, significa que E1 participa 4 vezes em S1 e 3 vezes em S2, ou, a cadeia que parte de E2, passa pelos conectores 16,1 e 2, significa que E2 participa 16 vezes em S1, 1 vez em S2 e 2 vezes em S3.

A figura (Fig. 2.15), a seguir, mostra a arquitetura do sistema DBGT Data Base Task Group:

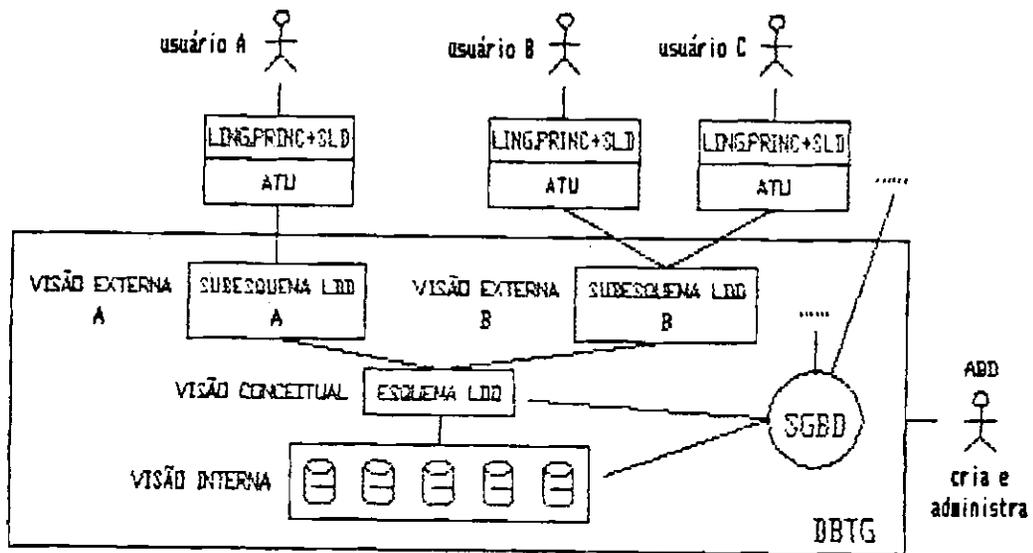


Fig. 2.15 - Arquitetura de um sistema DBT.

Existem 3 linguagens de representação:

1. Linguagem de descrição de dados: Esquema de linguagem de descrição de dados ESQUEMA LDD
2. Sublinguagem de descrição de dados: Subesquema de linguagem de descrição de dados SUBESQUEMA LDD
3. Linguagem de manipulação de dados: LMD

A visão conceitual é definida pelo ESQUEMA LDD, o qual consiste de definições dos vários tipos de registros do banco de dados, dos itens de dados que estes contém e os conjuntos nos quais estes são agrupados. Conjunto é o meio de representação dos relacionamentos ou tipos de relacionamentos.

A estrutura de armazenamento (visão interna) do banco de dados é descrita pelo esquema de armazenamento escrito em uma linguagem de descrição de armazenamento de dados LDAD.

A visão externa é definida por um subesquema SUBESQUEMA, o qual consiste de uma especificação sobre quais os tipos de registros esquema interessam ao usuário, quais os itens de dados esquema que o usuário deseja ver nesses registros e que relacionamentos esquema (conjunto) interligando esses registros o usuário deseja considerar.

Isto implica em que todos os outros tipos de registros, itens de dados e conjuntos fiquem automaticamente excluídos.

Os usuários são programadores de aplicações que usam qualquer linguagem comum estendida para incorporar a LMD do DBGIT.

Cada programa de aplicação invoca o subesquema correspondente. Esta invocação fornece a definição da área de trabalho do usuário, ATU. A ATU contém uma localização distinta para cada tipo de registro e portanto para cada tipo de dado definido no subesquema. O programa pode referenciar a estas localizações de itens de dados e registros por nomes definidos no subesquema.

2.3 - CONCLUSÃO E ESCOLHA DE UMA ESTRUTURA DE BANCO DE DADOS

Até aqui apresentou-se uma visão global sobre banco de dados, dada a necessidade de escolher um modelo de estrutura de dados a ser usado com entidades gráficas no sistema CAD, e que melhor se adapte à estrutura de armazenamento e manipulação de dados, tais como busca, inserção, atualização e remoção.

Lembrando que o objetivo do trabalho é a construção de um sólido através da composição de outros sólidos mais simples, seguindo um processo de construção hierárquica, a estrutura de dados usada pelo modelador geométrico pode ser manipulado de duas maneiras:

1. Por processo de síntese

No qual a partir de sólidos simples (nível mais baixo e gerados por varredura) e usando operações booleanas (união, intersecção e diferença) e/ou de transformação (translação, rotação ou mudança de escala) se chegue ao sólido final desejado.

2. Por processo de análise

Onde dada a necessidade de alterar um dos sólidos componentes do sólido final seja necessário decompô-lo descendo hierarquicamente na estrutura até o nível da alteração para logo voltar em um processo de síntese e obter um novo sólido final.

Levando em consideração estas características foi desenvolvido em um sistema CAD um software de alto nível (modelador geométrico) que implementa uma interface amigável através de menus entre o projetista CAD e a biblioteca gráfica residente, de tal maneira que o projetista pode usar de uma maneira fácil todos os recursos gráficos que o possibilitem modelar um sólido a partir de sólidos mais simples construídos por varredura ou usando todas as operações de combinação e transformação.

O software armazena todo o processo de construção do sólido final em um banco de dados com estrutura de dados hierárquica, capaz de descrever de uma maneira completa e não ambígua cada um dos sólidos componentes e cada entidade gráfica manipulada pelo projetista no processo.

O sistema utilizado no trabalho foi o sistema CAD/CAM Computervision's Designer V-X System e a linguagem usada foi a linguagem VARPRO 2, a qual é própria do sistema.

A seguir discute-se cada uma das abordagens de estrutura de dados e justifica-se o modelo de estrutura de dados seguido na implantação:

1. Sobre o modelo relacional

Serviu apenas como exemplo para a descrição da estrutura de dados através das tabelas relacionais. Dadas as limitações da linguagem usada (própria do sistema), que apesar de ser uma linguagem estruturada não suporta estruturas dinâmicas e portanto impossibilita o uso de ponteiros, a estrutura de dados definida é estática.

Porém, na manipulação de dados este modelo não serviu como base nesta implementação, porque o relacionamento entre as entidades é horizontal e no modelamento dos sólidos existe uma hierarquia de construção onde o relacionamento é sempre vertical seja em um processo de síntese ou de análise.

Este modelo poderia ser usado após terem sido modelados vários sólidos finais e existisse a necessidade de relacionar características entre eles, tais como cor, número de sólidos participantes na composição de cada sólido, quantidade de memória ocupada, etc., basicamente dados "administrativos".

No trabalho, o modelador geométrico está implementado para manipular apenas um sólido de cada vez.

2. Sobre o modelo hierárquico

O modelador geométrico implementado tem suas raízes neste modelo, pois este modelo tem uma estrutura de armazenamento hierárquica vertical, que possibilita relacionamentos 1:N com estrutura em árvore, o que é semelhante à estrutura de dados implantada.

Neste caso N é igual a 2 descrevendo uma árvore binária onde a raiz é o sólido final e onde cada nó representa um sólido de um determinado nível, composto pela combinação de dois sólidos de um nível hierárquico anterior. As folhas da árvore são sólidos gerados por varredura.

Apesar da linguagem usada não suportar estruturas dinâmicas nem o uso de ponteiros, dificultando operações em árvore, o software simula a estrutura dinâmica usando estruturas estáticas.

Isto é, usam-se estruturas estáticas com campos que representam ponteiros que definem outras estruturas estáticas, as quais também possuem campos que representam ponteiros a outras estruturas e assim por diante.

Este método permite a manipulação de dados hierarquicamente tanto no processo de síntese como no de análise.

Embora a estrutura de dados seja descrita de um modo relacional, a manipulação de dados do software na inserção, busca, atualização e remoção de dados tem características totalmente hierárquicas.

3. Sobre o modelo em rede

Na verdade o modelo hierárquico é um subconjunto deste modelo, o qual é um modelo mais extenso. Na estrutura de dados do modelador geométrico implantado, não existiu a necessidade de abordá-lo.

Este modelo agrupa relacionamentos em uma cadeia que se inicia no dono do relacionamento visitando todos seus membros e voltando ao dono; portanto o modelo em rede não é aplicável à estrutura de dados implantada, pois nesta estrutura a interatividade é sempre vertical excluindo relacionamentos horizontais e não obedece a nenhum caminho cíclico.

Todo sólido complexo pode ser composto (ou decomposto) a partir de (ou em) sólidos mais simples que pertencem a um nó ou folha da árvore, onde não existe nenhum relacionamento horizontal entre os nós, ou até mesmo entre as folhas.

Como o modelo representa uma árvore binária, um nó sempre terá apenas um pai e não mais de um como o modelo em rede suporta.

Portanto, a estrutura de dados do modelador geométrico implantado é descrita usando o modelo relacional, porém, esta estrutura é manipulada seguindo o modelo hierárquico. O modelo em rede não foi usado.

2.4 - PROPOSTA DE UMA ESTRUTURA DE BANCO DE DADOS GRÁFICO

Nesta parte descreve-se a estrutura de dados hierárquica do modelador geométrico implantado no sistema.

A estrutura de dados do modelador geométrico é uma estrutura hierárquica onde sólidos mais complexos são definidos a partir de sólidos mais simples pré-definidos em um nível de hierarquia menor.

A estrutura é composta por uma árvore binária onde as folhas da árvore são sólidos simples gerados pela varredura rotacional ou translacional de um contorno fechado (figura 2D composta a partir de figuras 1D) em torno de um eixo de rotação ou ao longo de um eixo de translação.

Os nós da árvore representam operações booleanas entre sólidos: união, intersecção e diferença, ou operações de transformação de um sólido: rotação, translação ou mudança de escala, e a raiz da árvore é o sólido final a ser modelado.

Deve ser notado que é empregado o termo "sólido" ao invés de objeto tridimensional, devido a que o sistema gera sólidos dos quais não só se tem informação de seus contornos mas também do seu interior e características, tais como massa e volume, como um sólido do mundo real. Assim, todos os níveis de hierarquia contêm sólidos.

A hierarquia imposta na estrutura de armazenamento obedece a 3 níveis, sendo o último nível subdividido em 2 subníveis: níveis 0,1,2 e 3, que representam instâncias da árvore binária.

O conceito de nível é melhor explicado no final deste subcapítulo quando é discutido o conceito de agregação e generalização.

A hierarquia imposta na estrutura de armazenamento é a seguinte:

1. Nível 0 ou contorno fechado

E o nível mais baixo da hierarquia determinado por uma figura 2D (ex: círculo) ou por um contorno fechado construído a partir de figuras 1D. A varredura rotacional ou translacional deste contorno gera um sólido que é a folha da árvore ou nível 1.

O projetista pode escolher para construir um contorno fechado as seguintes entidades gráficas:

- reta
- arco
- círculo
- B-Spline

2. Sólido primitivo ou nível 1

É o nível 1 ou folha da árvore. Estes são gerados no nível 0 por varredura e são usados, através de composição booleana, para gerar sólidos mais complexos. Como exemplos podem ser citados:

- usando varredura translacional, pode ser gerado um cubo usando como contorno um quadrado, ou, um cilindro usando como contorno um círculo;
- usando varredura rotacional, pode ser gerada uma esfera usando um semi-círculo como contorno (composto de um arco mais uma reta), um cilindro usando como contorno um retângulo, ou, um cone usando como contorno um triângulo.

3. Sólido transformado ou nível 2

Representa o nível 2 ou um nó da árvore. É um nível intermediário entre o sólido primitivo e o sólido final ou raiz da árvore. É definido pela transformação de corpo rígido de um sólido.

As transformações podem ser: rotação, translação ou mudança de escala (redução ou ampliação). O sólido a ser transformado pode ser:

- um sólido primitivo
- um sólido combinado
- um sólido transformado

4. Sólido combinado ou nível 2

Assim como o sólido transformado, este também representa o nível 2 ou um nó da árvore. É definido pela combinação booleana de dois sólidos de um nível hierárquico mais baixo.

As operações booleanas podem ser: união, intersecção e diferença. Este sólido pode ser composto:

- combinando-se 2 sólidos combinados
- combinando-se 2 sólidos transformados
- combinando-se 1 sólido combinado com 1 sólido transformado

5. Sólido final ou nível 3

É o sólido final modelado ou a raiz da árvore. Tem um nome associado que o identifica no banco de dados. Representa o nível mais alto do nível 2. Pode ser:

- um sólido combinado
- um sólido transformado
- um sólido primitivo

Todos estes sólidos fazem parte do modelo geométrico e podem ser identificados isoladamente no banco de dados para ser posicionados em qualquer ponto do sistema de coordenadas pois cada um dos sólidos está descrito totalmente na estrutura de dados armazenada.

Usando a notação BNF (Backus Normal Form) define-se a sintaxe de cada elemento do modelo geométrico:

(sól_final) := (sól_combi)|(sól_trans)|(sól_primi)

(sól_combi) := (sól_trans)(combinação)(sól_combi)|
(sól_combi)(combinação)(sól_trans)|
(sól_trans)(combinação)(sól_trans)|
(sól_combi)(combinação)(sól_combi)

(sól_trans) := (transformação)(sól_trans)|
(transformação)(sól_combi)|
(transformaçãI)(sól_primi)

(sól_primi) := (varredura)(contorno)

(contorno) := (reta)|(arco)|(círculo)|(B-Spline)

(combinação) := (união)|(intersecção)|(diferença)

(transformação):= (translação)|(rotação)|(escala)

(varredura) := (rotacional)|(translacional)

Portanto o processo de construção de um sólido composto pode ser constituído de várias operações tanto de transformação como de combinação.

As entidades gráficas utilizadas (reta, arco, círculo e BSpline), as operações de varredura, transformação e combinação, e o sombreado, são rotinas gráficas disponíveis na biblioteca gráfica do sistema CAD.

Portanto o software desenvolvido chama estas rotinas para fazer a interface entre o projetista e a construção do modelo geométrico desejado, guardando as informações em um banco de dados.

Foram implantadas duas abstrações de dados chamadas AGREGAÇÃO e GENERALIZAÇÃO, as quais facilitam a implementação da estrutura hierárquica.

A discussão sobre agregação e generalização a seguir apresentada deve ser acompanhada com a figura (Fig. 2.16) abaixo para um melhor entendimento:

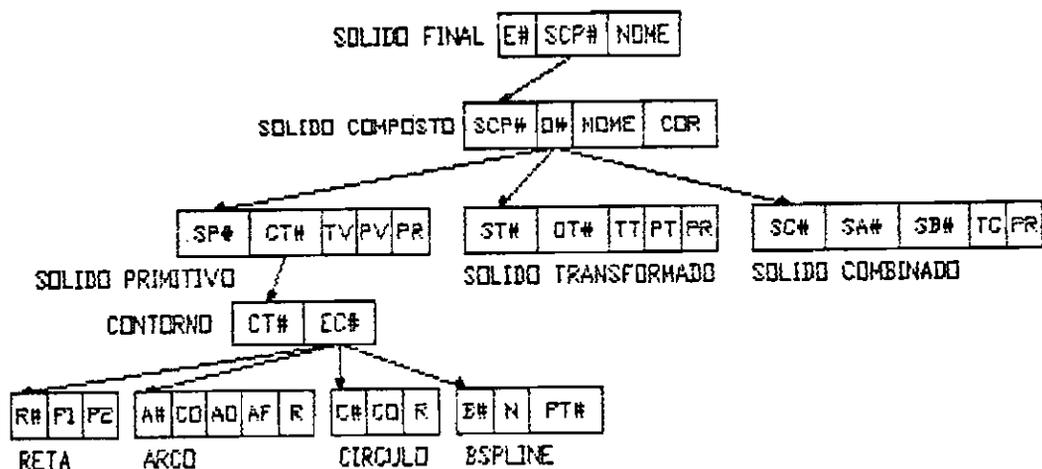


Fig. 2.16 - Modelo da estrutura de armazenamento do modelador geométrico.

O tipo de abstração chamado "agregação" permite relacionar vários objetos como um único objeto de mais alto nível. Este objeto é conhecido como objeto agregado. Basicamente segue uma associação vertical. O tipo registro ("record") em Pascal é uma implementação deste tipo.

Na estrutura de dados do modelador geométrico, os sólidos combinado, primitivo e transformado foram agregados como um sólido de mais alto nível em um registro chamado sólido composto, que é um registro que possui uma chave (campo) única que o identifica SCP# (campo apontado por um ponteiro do campo SCP# do sólido final), e outro campo O#, que aponta a um destes três registros dos sólidos agregados.

Na verdade, o sólido composto é uma lista dinâmica de registros onde cada registro representa uma instância de sólido composto ou um nó da árvore binária que associa dois nós de nível hierárquico menor (que pode ser um sólido combinado, transformado ou primitivo).

Assim, o sólido final, que também é um registro com um campo SCP#, ponteiro apontando o último registro da lista e que representa o objeto agregado de mais alto nível, não precisa necessariamente apontar o último registro desta lista mas sim à qualquer uma das suas instâncias. Pode-se portanto acessar qualquer um destes objetos agregados do banco de dados várias vezes com diferentes nomes.

Outro objeto agregado implementado é o registro contorno o qual tem um campo EC# que representa um ponteiro a uma lista dinâmica de vários registros que podem ser os registros de uma reta, arco, cilindro ou B-Spline.

A agregação de todos os registros desta lista dinâmica determina o registro contorno representativo de um contorno fechado, através do qual uma operação de varredura determina o sólido primitivo.

A agregação do registro sólido composto determina o nível 2 da hierarquia implantada. O objeto agregado de mais alto nível ou qualquer instância deste nível, apontado pelo sólido final, representa o nível 3 da hierarquia implantada.

No software desenvolvido o sólido final representa sempre a raiz da árvore ou objeto agregado de mais alto nível.

O tipo de abstração chamado "generalização" é um conjunto de objetos, com propriedades semelhantes, que pode ser visto como um objeto de mais alto nível com detalhes suprimidos. Este objeto é conhecido como objeto generalizado. Basicamente, é uma associação horizontal, e o tipo escalar ("scalar type") em Pascal o representa.

Na estrutura de dados definida, o campo O# do registro agregado sólido composto, que representa um ponteiro a outro sólido agregado, na verdade é uma generalização porque pode apontar um sólido combinado, um transformado ou um primitivo.

O campo EC# do registro agregado contorno também é uma generalização porque pode apontar qualquer um dos registros reta, arco, círculo ou B-Spline.

A seguir é dada uma explicação detalhada de cada registro do modelo da estrutura de dados mostrado na figura (Fig. 2.16) anterior:

SÓLIDO FINAL

E# : número de identificação
SCP# : ponteiro ao sólido composto
NOME : nome do sólido final

SÓLIDO COMPOSTO

SCP# : número de identificação
O# : ponteiro ao topo de uma lista dinâmica
que pode ser um registro de sólido
combinado, transformado ou primitivo
NOME : nome do sólido composto
COR : cor do sólido

SÓLIDO COMBINADO

SC# : número de identificação
SA# : ponteiro à um sólido combinado, transformado ou
primitivo que participa na operação booleana
SB# : ponteiro ao outro sólido que participa na
operação, pode ser um sólido combinado,
transformado ou primitivo
TC : tipo de combinação booleana
PR : ponto de referência, ponto no qual o sólido é
posicionado no espaço tridimensional

SÓLIDO TRANSFORMADO

ST# : número de identificação
OT# : ponteiro à um sólido combinado, transformado ou
primitivo, a ser transformado
TT : tipo de transformação de corpo rígido
PT : parâmetros de transformação
PR : ponto de referência

SÓLIDO PRIMITIVO

SP# : número de identificação
CT# : ponteiro à um contorno
TV : tipo de varredura
PV : parâmetros de varredura
PR : ponto de referência

CONTORNO

CT# : número de identificação
EC# : elementos do contorno, é um ponteiro ao topo de uma lista dinâmica de registros de reta, arco, círculo e B-Spline, que determinam um contorno fechado

RETA

R# : número de identificação
P1 : ponto inicial da reta
P2 : ponto final da reta

ARCO

A# : número de identificação
CO : centro do arco
AO : ângulo inicial do arco
AF : ângulo final do arco
R : raio do arco

CÍRCULO

C# : número de identificação
CO : centro do círculo
R : raio do círculo

BSPLINE

B # : número de identificação
N : número de pontos da curva B-Spline
PT# : ponteiro ao topo de uma lista dinâmica de pontos que determinam a curva B-Spline

A forma descrita acima é a considerada para a implantação dos níveis de hierarquia, da sintaxe de cada entidade gráfica e da representação gráfica do modelo de estrutura de dados usada pelo modelador geométrico.

A seguir é dado um exemplo de como poderia ser implementada a estrutura de dados do modelador geométrico em uma linguagem de alto nível, no caso a linguagem Pascal.

Cabe ressaltar que o sistema usado para implantação permite o uso de linguagens de alto nível como Pascal e Fortran; entretanto estas linguagens não têm acesso às rotinas gráficas residentes na biblioteca gráfica do sistema.

Existe porém uma linguagem própria do sistema dedicada exclusivamente a aplicativos CAD, que tem acesso a todas as rotinas gráficas residentes no pacote gráfico CADD5 4X, assim como permite usar todos os comandos do sistema operacional OS do sistema, esta linguagem é a VARPRO 2.

É este o principal motivo pelo qual o software desenvolvido ter sido implementado em VARPRO 2. E precisamente uma das limitações desta linguagem é que apesar de ser uma linguagem estruturada, não permite definir estruturas dinâmicas.

Assim, não é possível o uso de ponteiros como em Pascal e o software escrito em VARPRO 2 simula a estrutura dinâmica a partir de estruturas estáticas, os registros e ponteiros são implementados usando vetores alfanuméricos e arquivos.

Devido ao fato da linguagem não ser transportável a outro sistema, por ser exclusiva do mesmo, apresenta-se a implementação na linguagem Pascal, a qual permite definir estruturas dinâmicas e por ser uma linguagem universalmente conhecida, permitiria a transportabilidade do modelo a outro sistema.

```
PROGRAM BANCO_DADOS_GRAFICO;
```

```
TYPE
```

```
PTO = RECORD
```

```
    X,Y,Z : REAL;
```

```
END;
```

```
PONTOS = RECORD
```

```
    PONTO: PTO;
```

```
    POIN : ^PONTOS;
```

```
END;
```

```
RETA = RECORD
```

```
    R#    : INTEGER;
```

```
    P1,P2 : PTO;
```

```
END;
```

ARCO = RECORD

A# : INTEGER;
CO : PTO;
AO,AF,R : REAL;

END;

CIRCULO = RECORD

C# : INTEGER;
CO : PTO;
R : REAL;

END;

BSPLINE = RECORD

B#,N : INTEGER;
PT# : ^PONTOS;

END;

TIPO_ENTIDADE = (RETA,ARCO,CIRCULO,BSPLINE);

LISTA_CONT = RECORD

POIN : ^LISTA_CONT;
CASE TE : TIPO_ENTIDADE OF
RETA : (RETA : RETA);
ARCO : (ARCO : ARCO);
CIRCULO : (CIRCULO : CIRCULO);
BSPLINE : (BSPLINE : BSPLINE);

END;

CONTORNO = RECORD

CT# : INTEGER;
EC# : ^LISTA_CONT;

END;

```

EIXO = RECORD
        EO,E1 : PTO;
    END;

ROTACIONAL = RECORD
        EIXO   : EIXO;
        AO,AF  : REAL;
    END;

TRANSLACIONAL = RECORD
        EIXO   : EIXO;
        ALTURA : REAL;
    END;

TIPO_VARREDURA = (ROTACIONAL,TRANSLACIONAL);

SOL_PRIMI = RECORD
        SP# : INTEGER;
        CT# : ^CONTORNO;
        PR  : PTO;
        CASE TV : TIPO_VARREDURA OF
            ROTACIONAL:( PV: ROTACIONAL);
            TRANSLACIO:( PV: TRANSLACIONAL);
        END;
    END;

ROTACAO = RECORD
        RX,RY,RZ : REAL;
    END;

TRANSLACAO = RECORD
        TX,TY,TZ : REAL;
    END;

```

```

ESCALA = RECORD
    SX,SY,SZ : REAL
END;

TIPO_SOLIDO = (SOL_COMBI,SOL_TRANS,SOL_PRIMI);

TIPO_TRANSFORMACAO = (ROTACAO,TRANSLACAO,ESCALA);

APONTA_SOLIDO = RECORD
    CASE OAP# : TIPO_SOLIDO OF
        SOL_COMBI : (SOL_COMBI:"SOL_COMBI ");
        SOL_TRANS : (SOL_TRANS:"SOL_TRANS ");
        SOL_PRIMI : (SOL_PRIMI:"SOL_PRIMI ");
    END;

SOL_TRANS = RECORD
    ST : INTEGER;
    OT : APONTA_SOLIDO;
    PR : PTO;
    CASE TT : TIPO_TRANSFORMACAO OF
        ROTACAO : ( PT: ROTACAO );
        TRANSLACAO:( PT: TRANSLACAO);
        ESCALA : ( PT: ESCALA );
    END;

TIPO_COMBINACAO = (UNIAO,INTERSECAO,DIFERENCA);

SOL_COMBI = RECORD
    SC# : INTEGER;
    SA#,SB# : APONTA_SOLIDO;
    TC : TIPO_COMBINACAO;
    PR : PTO;
END;

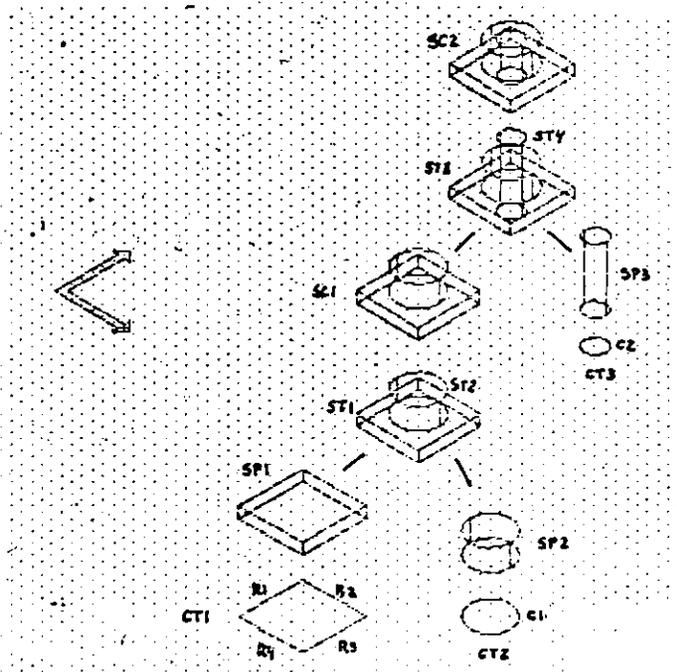
```

```
SOL_COMPOSTO = RECORD
    SCP#      : INTEGER;
    O#        : APONTA_SOLIDO;
    NOME,COR  : STRING [10];
    POIN      : ^SOL_COMPOSTO;
END;
```

```
SOLIDO_FINAL = RECORD
    E#  : INTEGER;
    SCP# : ^SOL_COMPOSTO;
    NOME : STRING [10];
END;
```

```
VAR
```

A figura (Fig. 2.17) mostrada a seguir, é um exemplo de como um sólido final é armazenado no banco de dados implementado:



RETA				ARCO					PONTOS		
R#	P1	P2		AB	CO	RO	AF	R	X1	Y1	Z1
R1	1	2							X1	Y1	Z1
R2	2	3							X2	Y2	Z2
R3	3	4							X3	Y3	Z3
R4	4	5							X4	Y4	Z4

CIRCULO			BSPLINE					CONTORNO					
C#	CO	R	B#	N#	P#	CT#	EN		X5	Y5	Z5		
C1	6	R1				CT1	P1	R2	R3	R4	X5	Y5	Z5
C2	11	R2				CT2	C1				X6	Y6	Z6
						CT3	C2				X7	Y7	Z7

SOLIDO PRIMITIVO					SOLIDO COMPOSTO				SOLIDO FINAL		
SP#	CT#	T	PU	BP	SCP#	GN	MGRE	COR	EM	SCP#	MGRE
SP1	CT1	T		5	SCP4	SP1			E1	SCP9	EXEMPLO
SP2	CT2	T		7	SCP2	SP2					
SP3	CT3	T		12	SCP3	ST1					
					SCP4	ST2					
					SCP5	SC1					
					SCP6	SP3					
					SCP7	ST2					
					SCP8	ST4					
					SCP9	SP2					

SOLIDO TRANSFORM					SOLIDO COMBINADO			
ST#	SP#	T	PT	BP	SC#	SC#	TC	BP
ST1	SP1	T		8	SC1	ST1	U	18
ST2	SP2	T		9	SC2	ST2	D	15
ST3	SP1	T		13				
ST4	SP3	T		14				

Fig. 2.17 - Representação de um sólido no banco de dados implantado.

CAPÍTULO 3

MODELAMENTO GEOMÉTRICO

3.1 - INTRODUÇÃO

Este capítulo enfoca o conceito de modelamento geométrico pois é a base fundamental para o entendimento da geometria das entidades que fazem parte da biblioteca gráfica do sistema assim como as sub-rotinas disponíveis.

Estas entidades gráficas, que podem ser formas geométricas 2D ou 3D (bidimensionais ou tridimensionais, respectivamente), são geometrias parametrizadas, isto é, definidas por equações paramétricas, as quais serão também apresentadas neste capítulo.

3.1.1 - DEFINIÇÃO DE MODELAMENTO GEOMÉTRICO

Surgiu nos anos 70 com o desenvolvimento da Computação Gráfica e a tecnologia CAD/CAM.

Consiste de uma coleção de métodos usados para definir a forma e outras características geométricas de um objeto real, proporcionando uma descrição matemática precisa ou um modelo. A combinação de ferramentas matemáticas com a complexidade potencial do modelo requer o poder de processamento de um computador.

O modelamento geométrico está associado ao modelamento de sólidos pois se aplica a formas e funções de qualquer dimensão.

Entre as áreas de aplicação podem ser citadas:

1. CAD/CAM
2. Computação Gráfica
3. Arte por computador
4. Animação por computador
5. Visão por computador
6. Robótica

O modelamento geométrico, divide-se em:

1. Geometria Paramétrica: curvas, superfícies e sólidos
2. Modelamento de Sólidos: construção e análise
3. Aplicações

Um Sistema de Modelamento de Sólidos consiste das seguintes partes:

1. Software de modelamento geométrico
2. Computador
3. Interface com o usuário (interface gráfica)
4. Banco de Dados para armazenar o modelo
5. Superfície de exibição ("display") de saída gráfica

3.1.2 - FORMAS GEOMÉTRICAS NOMEÁVEIS E NÃO NOMEÁVEIS

Os elementos nomeáveis são aqueles da geometria clássica tais como planos, esferas, círculos, elipses, parábolas, retas, etc.

Os elementos não nomeáveis são as figuras em três dimensões geradas por sistemas CAD/CAM as quais são não clássicas.

3.1.3 - EQUAÇÕES PARAMÉTRICAS

Para expressar elementos geométricos matematicamente, podem ser usadas equações paramétricas ou não paramétricas.

As equações não paramétricas são expressas de forma explícita ou implícita.

Uma equação explícita não paramétrica da forma $y=f(x)$ não conseguiria expressar uma curva fechada pois existe um só valor de y para cada x . Este tipo de função é chamada de unívoca.

Isto poderia ser solucionado usando uma equação implícita não paramétrica da forma $f(x,y)=0$, porém existe a limitação de os elementos não paramétricos, x e y , serem dependentes do sistema de coordenadas escolhido.

Assim mesmo, são úteis as definições de propriedade intrínseca e propriedade extrínseca.

Uma propriedade intrínseca é aquela que depende somente da figura e não do sistema de referência. Uma propriedade extrínseca depende somente do sistema de referência e não da figura.

Por exemplo, o fato de um retângulo ter 4 lados iguais é intrínseco ao retângulo, porém o fato de retângulo ter 2 lados verticais é extrínseco pois o sistema de referência é necessário para determinar qual direção é vertical.

Além do mais, não é possível expressar figuras no modelamento geométrico usando equações explícitas ou implícitas não paramétricas, pelos seguintes motivos:

- As formas da maioria dos objetos são intrinsecamente independentes de qualquer sistema de coordenadas.
- Na determinação de uma curva ou superfície pela união de uma série de pontos, observa-se que é a relação entre os pontos quem determina a figura e não a relação destes pontos com algum sistema de coordenadas, portanto a escolha do sistema de coordenadas não deve afetar a figura.
- Qualquer sólido ou objeto fechado terá linhas verticais, tangentes ou planos com respeito a um sistema de coordenadas, e se este for dependente do sistema de coordenadas, implicaria em inclinações infinitas e propriedades matemáticas não bem definidas.
- As curvas e superfícies do modelamento geométrico são não planas e limitadas, tal que não podem ser representadas por funções não paramétricas.

Por estes motivos e outros relacionados com tempo de processamento e programação, a maneira predominante de representação de figuras no modelamento geométrico é feita usando-se equações paramétricas, a seguir explicadas:

por exemplo, a representação paramétrica de uma curva 2D pode ser expressa por duas funções

$$x = x(u) \quad e \quad y = y(u) , \quad (3.1)$$

em função de um parâmetro u . Assim, um ponto em tal curva é representado pelo vetor

$$\mathbf{p} = | x(u) \quad y(u) | , \quad (3.2)$$

um ponto em uma curva no espaço é representado pelo vetor

$$\mathbf{p} = | x(u) \quad y(u) \quad z(u) | , \quad (3.3)$$

e um ponto em uma superfície é representado pelo vetor

$$\mathbf{p} = | x(u,w) \quad y(u,w) \quad z(u,w) | , \quad (3.4)$$

considerando nestes exemplos a matriz

$$| \mathbf{i} \quad \mathbf{j} \quad \mathbf{k} | \quad (3.5)$$

As funções matemáticas que geram um conjunto de pontos definindo curvas, superfícies e outros elementos geométricos são equações paramétricas.

Veja as equações paramétricas da curva mostrada na figura (Fig. 3.1) abaixo:

$$x=3u^2 \quad y=u^3-u+1 \quad z=2u+3 \quad (3.6)$$

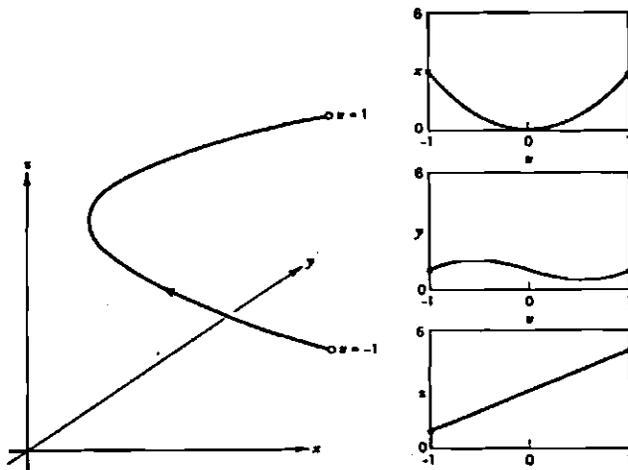


Fig. 3.1 - Uma curva paramétrica.

FONTE: Mortenson (1985), p. 24.

Na figura (Fig. 3.1) u é a variável paramétrica independente. Cada valor de u gera um valor específico de x, y e z e portanto um ponto na curva. Também tem-se outras curvas: $(x, u), (y, u)$ e (z, u) . Limitando o valor de u no intervalo $[0, 1]$ normaliza-se a variável paramétrica e determina-se as condições de contorno das curvas e superfícies.

3.2 - CURVAS

Apresenta-se aqui o conceito de curvas como base para o entendimento da geometria das curvas B-Spline as quais foram manipuladas como entidades na estrutura de banco de dados desenvolvida, cabe ressaltar que a reta também é um caso especial de curvas.

A representação de uma curva é dada através de suas matrizes de coeficientes algébricos e geométricos, já que os métodos matriciais permitem um poder de processamento maior.

3.2.1 - DEFINIÇÃO

Um segmento de curva é uma coleção de pontos limitados por dois pontos extremos cujas coordenadas são dadas por funções matemáticas contínuas de um parâmetro da forma

$$x=x(u) \quad y=y(u) \quad z=z(u) \quad (3.7)$$

A variável paramétrica u é limitada ao intervalo $[0,1]$. A curva é contornada por dois pontos extremos em $u=0$ e $u=1$. As coordenadas de qualquer ponto na curva paramétrica são os componentes do vetor $p(u)$.

Na figura (Fig. 3.2) abaixo ilustram-se os elementos vetoriais de uma curva paramétrica

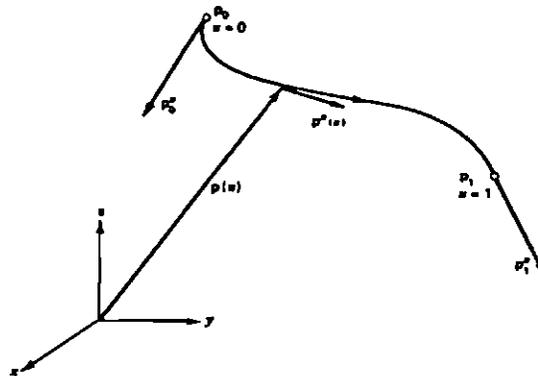


Fig. 3.2 - Elementos vetoriais de uma curva paramétrica.

FONTE: Mortenson (1985), p. 31.

Aqui $\mathbf{p}(u)$ é o vetor ao ponto $x(u), y(u), z(u)$ e $\mathbf{p}'(u)$ é o vetor tangente à curva no mesmo ponto e é achado derivando $\mathbf{p}(u)$, logo

$$\begin{aligned} \mathbf{p}(u) &= | x(u) \quad y(u) \quad z(u) | \\ \mathbf{p}'(u) &= d\mathbf{p}(u)/du \\ \mathbf{p}'(u) &= | x^u \quad y^u \quad z^u | , \end{aligned} \quad (3.8)$$

onde os vetores componentes são:

$$x^u = dx(u)/du \quad y^u = dy(u)/du \quad z^u = dz(u)/du \quad (3.9)$$

x^u , y^u , e z^u são as derivadas paramétricas. As relações entre as derivadas paramétricas e as derivadas ordinárias do espaço cartesiano são:

$$\begin{aligned}
dy/dx &= (dy/du)/(dx/du) \\
dy/dz &= (dy/du)/(dz/du) \\
dz/dx &= (dz/du)/(dx/du)
\end{aligned}
\tag{3.10}$$

Os vetores $p(0)$, $p^u(0)$, $p(1)$ e $p^u(1)$ são as condições de contorno. A curva é limitada pelo ponto $p(0)$ em $u = 0$ e pelo ponto $p(1)$ em $u=1$.

Em geral $p(u)$ não se restringe a uma curva 3D no espaço cartesiano, portanto

$$p(u) = | x_1(u) \quad x_2(u) \quad \dots \quad x_n(u) | \tag{3.11}$$

3.2.2 - FORMAS ALGÉBRICA E GEOMÉTRICA

A forma algébrica de um segmento de curva cúbica parametrizada (ou curva pc - "parametric cubic curve") é dada por:

$$\begin{aligned}
x(u) &= a_{3x}u^3 + a_{2x}u^2 + a_{1x}u + a_{0x} \\
y(u) &= a_{3y}u^3 + a_{2y}u^2 + a_{1y}u + a_{0y} \\
z(u) &= a_{3z}u^3 + a_{2z}u^2 + a_{1z}u + a_{0z}
\end{aligned}
\tag{3.12}$$

u é a variável independente restrita ao intervalo $[0,1]$ inclusive, determinando que a curva pc seja contornada. O conjunto único de 12 coeficientes constantes, chamados Coeficientes Algébricos, determinam uma única curva pc. Estes coeficientes determinam o tamanho, forma e posição no espaço da curva pc.

Podem ser representadas as 3 equações algébricas em uma forma vetorial:

$$\mathbf{p}(u) = \mathbf{a}_3 u^3 + \mathbf{a}_2 u^2 + \mathbf{a}_1 u + \mathbf{a}_0 \quad (3.13)$$

$$\begin{aligned} \text{tal que: } \mathbf{a}_3 &= \begin{vmatrix} a_{3x} & a_{3y} & a_{3z} \end{vmatrix} \\ \mathbf{a}_2 &= \begin{vmatrix} a_{2x} & a_{2y} & a_{2z} \end{vmatrix} \\ \mathbf{a}_1 &= \begin{vmatrix} a_{1x} & a_{1y} & a_{1z} \end{vmatrix} \\ \mathbf{a}_0 &= \begin{vmatrix} a_{0x} & a_{0y} & a_{0z} \end{vmatrix}, \end{aligned} \quad (3.14)$$

onde: $\mathbf{p}(u)$ é a posição vetorial de qualquer ponto na curva

$\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2$ e \mathbf{a}_3 são os vetores equivalentes dos coeficientes algébricos escalares.

Os coeficientes algébricos não oferecem um senso intuitivo da curva e portanto nem sempre maneira mais conveniente de controlar a forma da curva.

A forma geométrica vem solucionar este problema. Uma curva pc , pode ser definida em termos de condições nos seus pontos extremos dados pelos vetores $\mathbf{p}(0)$ e $\mathbf{p}(1)$, e seus correspondentes vetores tangentes

$$\mathbf{p}^u(0) = d\mathbf{p}(0)/du \quad \text{e} \quad \mathbf{p}^u(1) = d\mathbf{p}(1)/du \quad (3.15)$$

A forma geométrica é dada por

$$\mathbf{p}(u) = F_1(u)\mathbf{p}(0) + F_2(u)\mathbf{p}(1) + F_3(u)\mathbf{p}^u(0) + F_4(u)\mathbf{p}^u(1) \quad (3.16)$$

$$\text{ou: } \mathbf{p} = F_1\mathbf{p}_0 + F_2\mathbf{p}_1 + F_3\mathbf{p}_0^u + F_4\mathbf{p}_1^u \quad (3.17)$$

Aqui $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_0^u$ e \mathbf{p}_1^u são os Coeficientes Geométricos e os termos F representam as funções de Blending, os quais são dados por:

$$\begin{aligned}
F_1(u) &= 2u^3 - 3u^2 + 1 \\
F_2(u) &= -2u^3 + 3u^2 \\
F_3(u) &= u^3 - 2u^2 + u \\
F_4(u) &= u^3 - u^2
\end{aligned}
\tag{3.18}$$

A forma matricial é a forma mais compacta para representar uma curva pc e portanto operações geométricas são simples manipulações de matrizes.

Expressando matricialmente a forma algébrica:

$$p = UA \tag{3.19}$$

$$\text{onde: } U = \begin{vmatrix} u^3 & u^2 & u^1 & 1 \end{vmatrix} \tag{3.20}$$

$$A = \begin{vmatrix} a_3 & a_2 & a_1 & a_0 \end{vmatrix} \tag{3.21}$$

Analogamente para a forma geométrica, tem-se:

$$p = FB \tag{3.22}$$

$$\text{onde: } F = \begin{vmatrix} F_1 & F_2 & F_3 & F_4 \end{vmatrix} \tag{3.23}$$

$$B = \begin{vmatrix} p_0 & p_1 & p_0^u & p_1^u \end{vmatrix}^T \tag{3.24}$$

A é a matriz de coeficientes algébricos e B é matriz de coeficientes geométricos ou matriz das condições de contorno.

As equações a seguir, permitem a conversão entre as formas algébricas e geométricas:

$$A = MB \quad \text{e} \quad B = M^{-1}A \quad (3.25)$$

onde:

$$M^{-1} = \begin{vmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{vmatrix} \quad (3.26)$$

M é a matriz de transformação universal ou matriz de Hermite.

As curvas pc definidas pela sua forma geométrica (vetores e tangentes nos pontos extremos) são conhecidas como curvas de Hermite. Observe que as matrizes U, F e M são idênticas para todas as curvas pc, somente as matrizes A e B variam de curva para curva e são estas as que controlam a forma, tamanho e posição no espaço da curva pc.

3.2.3 - ESPAÇO OBJETO E ESPAÇO PARAMÉTRICO DA CURVA

O Espaço Objeto é o espaço tridimensional definido pelas coordenadas cartesianas x, y e z. É o espaço no qual o modelo geométrico é totalmente desenvolvido.

O Espaço Paramétrico de uma curva é o conjunto de espaços bidimensionais definidos pelas coordenadas (x,u), (y,u) e (z,u).

Pode-se decompor qualquer curva paramétrica em seus 3 componentes no espaço paramétrico. Os gráficos das curvas pc no espaço paramétrico são geralmente úteis para visualizar seu comportamento no espaço objeto.

A figura (Fig. 3.1) mostrada anteriormente ilustra estes dois tipos de espaço.

3.2.4 - LINHAS RETAS

Impondo a condição dos vetores tangentes serem função dos vetores dos pontos extremos, isto é:

$$p_0^u = (p_1 - p_0) \quad p_1^u = (p_1 - p_0) , \quad (3.27)$$

e substituindo estes valores na matriz de coeficientes geométricos B determina-se a forma geométrica para uma linha reta. Ou seja:

$$p = F \begin{vmatrix} p_0 & p_1 & (p_1 - p_0) & (p_1 - p_0) \end{vmatrix}^T \quad (3.28)$$

A figura (Fig. 3.3) abaixo, ilustra um exemplo de uma reta pc linear:

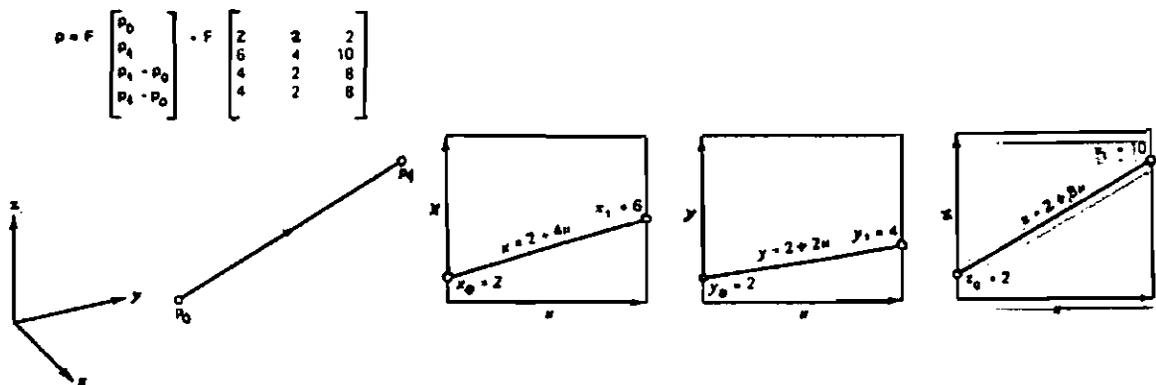


Fig. 3.3 - Reta pc.

FONTE: Mortenson (1985),
p. 73.

3.2.5 - CURVAS SPLINE

A curva Spline é uma curva suave aproximada que pode ser desenhada através de qualquer conjunto de n pontos que impliquem em uma curva suave. A variação de mudança da curva é gradual e não existem pontos de quebra.

Estes pontos são chamados pontos de controle, e dizer que a curva interpola estes pontos, quer dizer que a curva passa através de todos eles.

A curva Spline geralmente usada, é a curva plana que pode ser definida por uma função cúbica.

A figura (Fig. 3.4) abaixo é um exemplo de uma curva Spline:

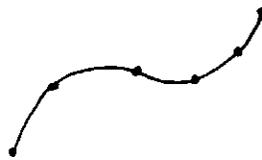


Fig. 3.4 - Curva Spline.

3.2.6 - CURVAS DE BEZIER

Algumas técnicas de definição de curvas interpolam um dado conjunto de pontos, o que significa que a curva produzida passa exatamente por sobre os pontos.

Um recurso alternativo define uma curva que somente se aproxima dos pontos dados. Este recurso é conhecido como curvas de Bézier, o qual permite controlar melhor a forma de uma curva, pois ao contrário de uma curva Spline interpolada onde a mudança de um ou mais pontos de interpolação podem produzir inflexões não esperadas, neste recurso basta mudar uns poucos parâmetros.

Os trabalhos de Bézier se relacionam com curvas cúbicas paramétricas e técnicas de interpolação de superfícies bicúbicas.

Bézier postulou que qualquer ponto no segmento de curva deve ser dado por uma função paramétrica da seguinte forma:

$$\mathbf{p}(u) = \sum_{i=0}^n \mathbf{p}_i f_i(u) \quad u \in [0,1] , \quad (3.29)$$

onde os vetores \mathbf{p}_i representam os $n+1$ vértices de um polígono característico ou também chamados pontos de controle.

A figura (Fig. 3.5) a seguir, ilustra este tipo de curva:

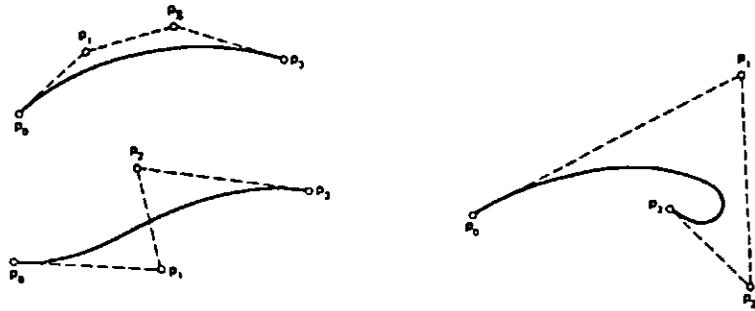


Fig. 3.5 - Curvas de Bezier.

FONTE: Mortenson (1985), p. 114.

As funções de blending $f_i(u)$ devem ter (entre outras) as seguintes propriedades:

- as funções devem interpolar o primeiro e último ponto de vértice, isto é, o segmento de curva deve começar em p_0 e terminar em p_n ;
- a tangente em p_0 deve ser dada por $p_1 - p_0$ e a tangente em p_n por $p_n - p_{n-1}$. Isto dá o controle direto da tangente à curva em cada extremo;
- as funções $f_i(u)$ devem ser simétricas com respeito a u e $(1-u)$. Isto significa que pode-se inverter a sequência dos pontos de vértice que definem a curva sem mudar a forma da curva, apenas muda a direção de parametrização.

Bézier usou a família de curvas chamadas polinômios de Bernstein para satisfazer estas condições, tal que:

$$p(u) = \sum_{i=0}^n p_i B_{i,n}(u) \quad u \in [0,1] \quad , \quad (3.30)$$

$$\text{onde: } B_{i,n}(u) = C(n,i)u^i(1-u)^{n-i} \quad (3.31)$$

$$C(n,i) = n!/(i!(n-i)!) \quad (3.32)$$

A Equação (3.30) é uma equação vetorial que pode ser escrita separadamente nas funções paramétricas x, y e z, como mostrado abaixo:

$$\begin{aligned} x(u) &= \sum_{i=0}^n x_i B_{i,n}(u) \\ y(u) &= \sum_{i=0}^n y_i B_{i,n}(u) \\ z(u) &= \sum_{i=0}^n z_i B_{i,n}(u) \quad , \end{aligned} \quad (3.33)$$

assim para 4 pontos, n=3 se tem:

$$p(u) = (1-u)^3 p_0 + 3u(1-u)^2 p_1 + 3u^2(1-u) p_2 + u^3 p_3 \quad , \quad (3.34)$$

esta curva está ilustrada na figura (Fig. 3.6) a seguir, observe que as curvas são tangentes as retas definidas por $p_1 - p_0$ e $p_3 - p_2$:

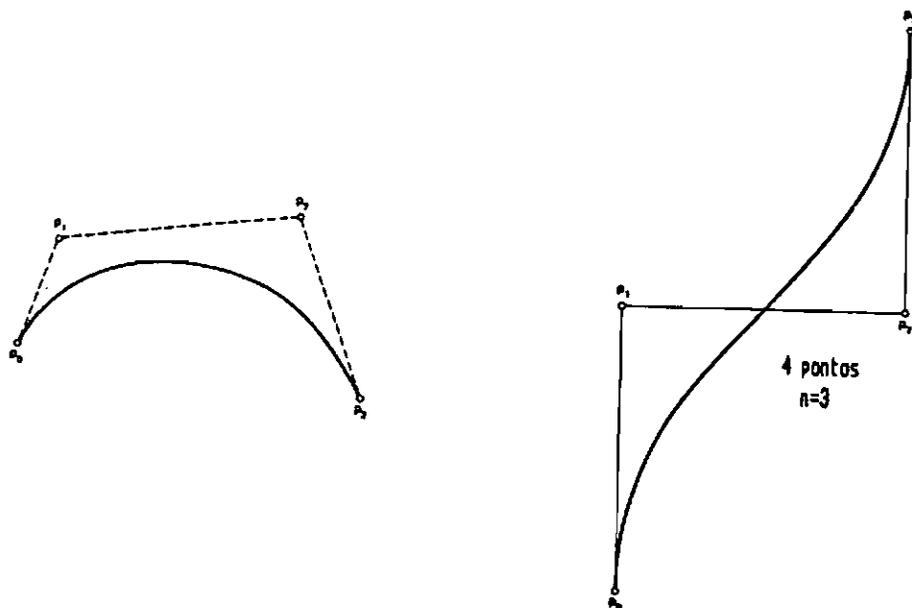


Fig. 3.6 - Curva cúbica de Bézier.

FONTE: Mortenson (1985), p. 116.

As funções de blending determinam o comportamento das curvas de Bézier.

Na figura (Fig. 3.7) a seguir, observam-se as quatro funções blinding que correspondem à curva de Bézier com quatro pontos de controle.

Estas curvas representam a influência que cada ponto de controle exerce na curva para vários valores de u . O primeiro ponto de controle, p_0 , correspondente a $B_{0,3}$, é o que mais influencia quando $u=0$, pois os outros pontos de controle são ignorados devido a que suas funções de blinding para $u=0$ são iguais a zero. A mesma situação ocorre para p_1 , p_2 , e p_3 , quando u vale $1/3$, $2/3$ e 1 respectivamente:

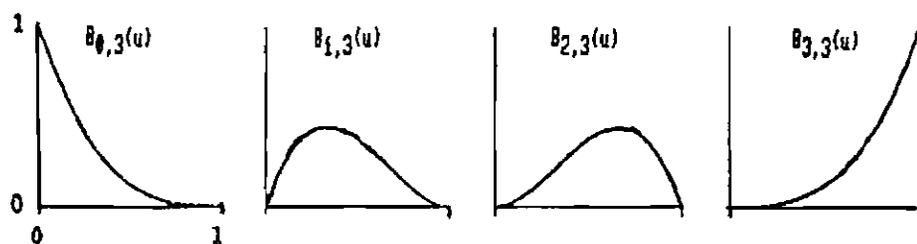


Fig. 3.7 - Funções de blending para $n=3$.

FONTE: Newman e Sproull (1979), p. 316.

3.2.7 - CURVAS B-SPLINE

A maioria das técnicas de definição de curvas não permitem um controle local da forma da curva.

Conseqüentemente, mudanças locais, como por exemplo, uma pequena mudança na posição de um ponto na curva Spline ou em um vértice do polígono característico da curva de Bézier, tendem a se propagar fortemente através da curva inteira. Isto é descrito como propagação global de mudança.

A curva B-Spline evita este problema usando um conjunto especial de funções blending as quais têm apenas influência local e dependem somente de pequenos pontos vizinhos.

Curvas B-Spline são semelhantes às curvas de Bézier no sentido de que um conjunto de funções de blending combinam os efeitos de $n+1$ pontos de controle p_i dados por

$$p(u) = \sum_{i=0}^n p_i N_{i,k}(u) \quad (3.35)$$

Esta equação é semelhante as curvas de Bézier; a diferença está na formulação das funções de blending $N_{i,k}(u)$.

Para as curvas de Bézier o número de pontos de controle determinam o grau das funções polinomiais de blending.

Porém, para as curvas B-Spline o grau destes polinômios é controlado pelo parâmetro k , o qual é geralmente independente do número de pontos de controle.

A figura (Fig. 3.8) a seguir, mostra as seis funções de blending para seis pontos de controle ($n=5$). O parâmetro k controla a ordem de continuidade da curva: na figura $k=3$. O parâmetro u varia em uma faixa maior do que as funções de Bézier: de 0 a $n-k+2$ na figura.

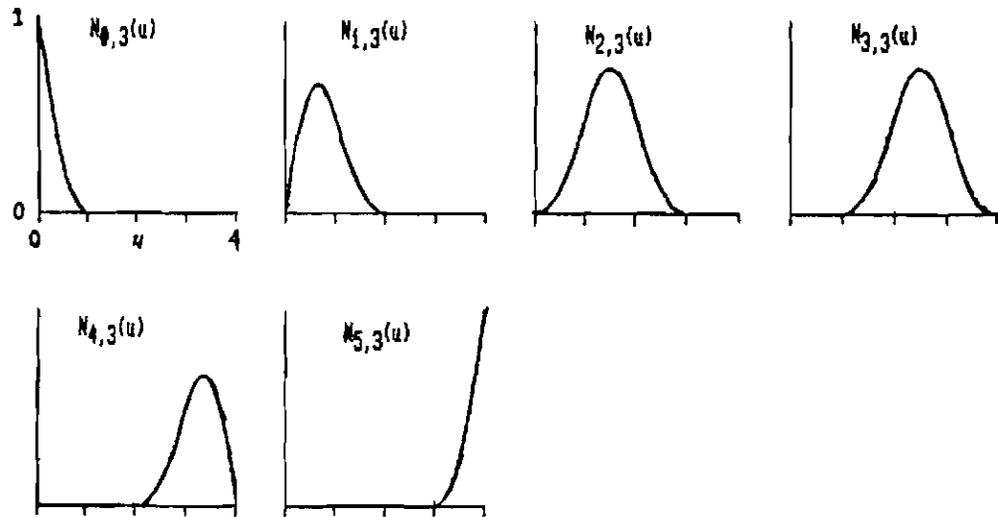


Fig. 3.8 - Funções de blending B-Spline para $n=5$, $k=3$.
 FONTE: Newman e Sproull (1979), p. 322.

A característica mais importante deste tipo de funções blending, é que elas são não nulas somente em uma determinada parte da faixa de valores do parâmetro u .

Por exemplo, o segundo ponto de controle influencia na forma da curva somente na faixa $0 \leq u \leq 3$, devido a que $N_{2,3}$ é zero para todos os outros valores de u .

Portanto, pelos menos três dos seis pontos de controle influenciam na forma local da curva (em um determinado ponto de controle).

As funções blending B-Spline de grau $k-1$ são definidas recursivamente por:

$$N_{i,1}(u) = \begin{cases} 1 & \text{se } t_i \leq u < t_{i+1} \\ 0 & \text{caso contrario} \end{cases} \quad (3.36)$$

$$N_{i,k}(u) = \frac{(u-t_i)N_{i,k-1}(u)}{(t_{i+k-1}-t_i)} + \frac{(t_{i+k}-u)N_{i+1,k-1}(u)}{(t_{i+k}-t_{i+1})},$$

onde k controla o grau ($k-1$) do polinômio resultante em u e portanto também controla a continuidade da curva. Os t_i são chamados valores "knot" e relacionam a variável paramétrica u com os pontos de controle p_i .

Para uma curva aberta, chamada curva B-Spline não periódica, devido a curva passar somente pelo primeiro e último ponto de controle, como mostra a figura (Fig. 3.9) a seguir, os valores de t_i são:

$$\begin{aligned} t_i &= 0 & \text{se } i < k \\ t_i &= i-k+1 & \text{se } k \leq i \leq n \\ t_i &= n-k+2 & \text{se } i > n, \end{aligned} \quad (3.37)$$

com:

$$\begin{aligned} 0 \leq i \leq n+k \\ 0 \leq u \leq n-k+2 \end{aligned}$$

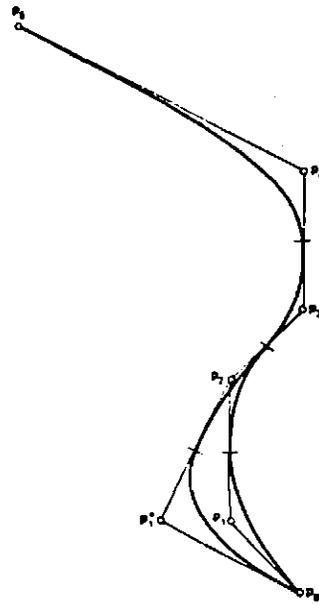


Fig. 3.9 - Curva B-Spline não periódica: $n=5$, $k=3$.
 FONTE: Mortenson (1985), p. 134.

As curvas B-Spline que modelam curvas fechadas, chamadas curvas B-Spline periódicas, tem a característica de não passar por nenhum ponto de controle.

Estas curvas, estão mostradas na figura (Fig. 3.10) a seguir.

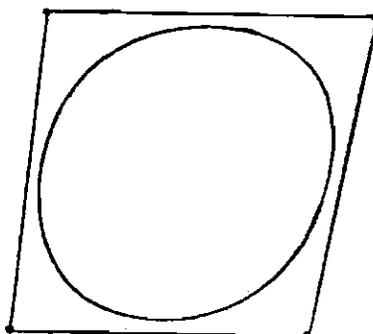


Fig. 3.10 - Curva B-Spline periódica com 4 pontos de controle ($k=4$).

FONTE: Newman e Sproull (1979), p. 324.

Para este tipo de curvas considera-se $t_i=i$ reduzindo todas as funções blending a uma só, dada por:

$$N_{i,k}(u) = N_{0,k}((u-i+n+1) \bmod (n+1)) \quad (3.38)$$

onde: $0 \leq u \leq n+1$

Na figura (Fig. 3.9) acima, observa-se que o ponto p_1 é movido a p_1^* afetando apenas dois segmentos da curva, isto é verdade pois cada segmento da curva B-Spline é influenciado por somente k pontos de controle e inversamente cada ponto de controle influencia somente em k segmentos de curva. Na figura $k=3$ e três pontos de controle afetam apenas dois segmentos.

É o controle local da forma da curva que dá à técnica B-Spline uma vantagem em relação a técnica Bézier, assim como a capacidade, de adicionar pontos de controle sem incrementar o grau da curva.

3.3 - SUPERFÍCIES

Inicialmente são introduzidas as superfícies paramétricas para logo mostrar como estas podem ser representadas usando retalhos bicúbicos ("bicubic patch") os quais através de suas matrizes algébricas e geométricas permitem métodos computacionais mais eficientes.

São apresentadas as superfícies plana, cilíndrica, esférica e de revolução.

3.3.1 - DEFINIÇÃO

O elemento matemático mais simples usado para modelar uma superfície é o retalho ("patch").

O retalho é uma coleção de pontos de uma curva limitada por dois pontos extremos (curva limitada) cujas coordenadas são dadas por funções matemáticas de valores únicos, contínuas e de dois parâmetros da forma:

$$x = x(u,w) \quad y = y(u,w) \quad z = z(u,w) \quad (3.39)$$

as variáveis paramétricas u e w estão confinadas aos intervalos $u, w \in [0,1]$.

Fixando uma das variáveis em um valor e fazendo variar a outra variável paramétrica no intervalo permitido, obtemos uma curva no retalho.

Assim se fixarmos a primeira variável, fazendo variar a segunda para todos seus valores e depois fixarmos a segunda fazendo variar a primeira, obtém-se no retalho uma grade paramétrica de duas famílias de curvas de um parâmetro tal que apenas uma curva de cada família passa através de cada ponto $p(u,w)$.

Observe a figura (Fig. 3.11) abaixo:

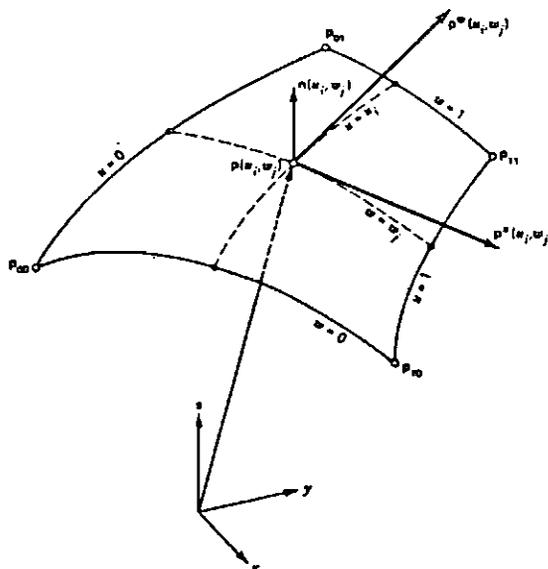


Fig. 3.11 - Retalho paramétrico de superfície.

FONTE: Mortenson (1985), p. 152.

Associado a cada retalho temos um conjunto de condições de contorno. As condições mais óbvias são os 4 pontos de esquina e as 4 curvas definindo os lados do retalho, isto é consequência da combinação dos 2 limites das duas variáveis paramétricas.

Determina-se os 4 pontos de esquina substituindo as quatro combinações de 0 e 1 em $p(u,w)$ para obter:

$$p(0,0), p(0,1), p(1,0) \text{ e } p(1,1).$$

Por outro lado, as curvas de contorno são funções de uma das duas variáveis paramétricas. Fixando uma das variáveis a seus valores limites enquanto a outra varia livremente, obtém-se as 4 funções das 4 curvas paramétricas de contorno:

$$p(0,u), p(u,1), p(0,w) \text{ e } p(1,w).$$

O exemplo mais simples de um retalho paramétrico de superfície é o plano. As equações abaixo representam um segmento retangular do plano xy :

$$x = (c-a)u+a \quad y = (d-b)w+b \quad z = 0 \quad (3.40)$$

Observe a figura (Fig. 3.12) a seguir. Aqui são mostradas as coordenadas paramétricas e as coordenadas x,y de cada ponto de esquina.

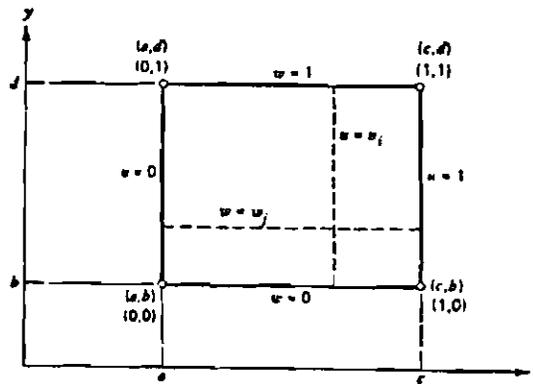


Fig. 3.12 - Coordenadas paramétricas e coordenadas xy de um plano.

FONTE: Mortenson (1985), p. 153.

As curvas de w constante são linhas retas paralelas ao eixo x . Estas curvas são funções de u . Analogamente, existem retas de u constante paralelas ao eixo y que são funções de w .

Outra superfície simples é a esfera, definida como o lugar geométrico de um ponto que se move a uma distância constante de um ponto fixo.

As equações paramétricas de uma esfera de raio r centrada em um ponto (x_0, y_0, z_0) são:

$$\begin{aligned}
 x &= x_0 + r \cos u \cos w & u &\in [-\pi/2, \pi/2] \\
 y &= y_0 + r \cos u \sin w & w &\in [0, 2\pi] \\
 z &= z_0 + r \sin u ,
 \end{aligned}
 \tag{3.41}$$

onde u é análogo à latitude e w à longitude, com ambos ângulos em radianos, isto é, a latitude representa um círculo gerado em um plano paralelo ao plano xy conforme w varia. As curvas de w constante são meridianos de longitude gerando semi-círculos conforme u varia.

Veja a figura (Fig. 3.13) a seguir:

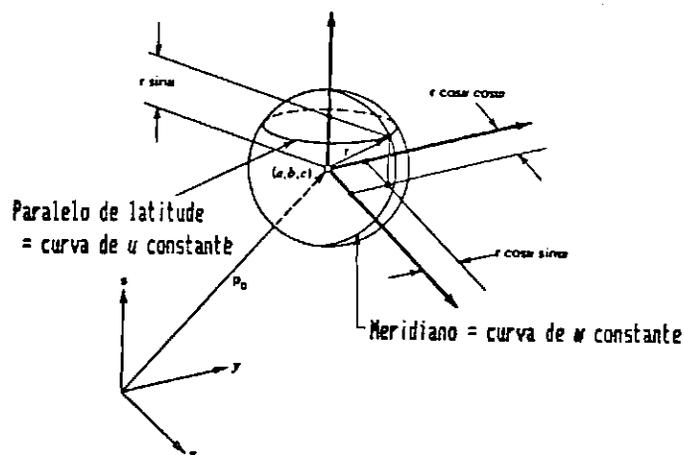


Fig. 3.13 - Esfera paramétrica.

FONTE: Mortenson (1985), p. 153.

Por último, na figura (Fig. 3.14) a seguir, tem-se uma superfície paramétrica de revolução, onde está mostrada apenas uma parte da superfície.

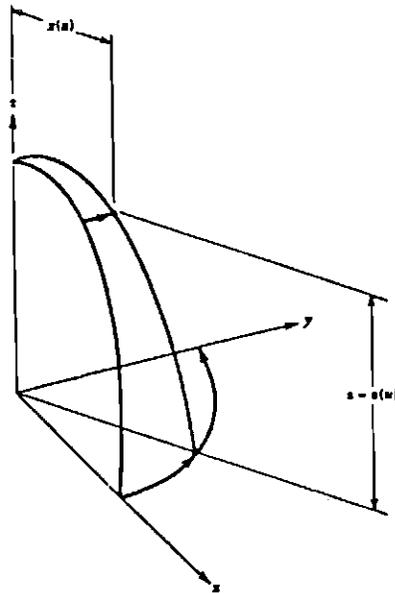


Fig. 3.14 - Superfície paramétrica de revolução.

FONTE: Mortenson (1985), p. 155.

Aqui, a curva definida pela função $x(u), z(u)$ é revolucionada em torno do eixo z , e tem como equações:

$$\begin{aligned}
 x &= x(u) \cos w & w &\in [0, 2\pi] \\
 y &= x(u) \sin w \\
 z &= z(u)
 \end{aligned}
 \tag{3.42}$$

3.3.2 - FORMAS ALGÉBRICA E GEOMÉTRICA

A forma algébrica de um retalho bicúbico é dado por:

$$p(u,w) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} u^i w^j \quad (3.43)$$

onde: $u, w \in [0,1]$

Os vetores a_{ij} são chamados os coeficientes algébricos da superfície.

A razão do termo "bicúbico" deve-se a que as variáveis paramétricas podem ser termos cúbicos. Assim como para curvas, as variáveis paramétricas são restritas ao intervalo $[0,1]$ inclusive, esta restrição determina que a superfície seja limitada (por 4 curvas) de uma maneira regular.

Expandindo a Equação (3.43) chega-se a um polinômio de 16 termos em u e w que definem todos os pontos da superfície. Esta equação é a forma algébrica de um retalho bicúbico.

$$\begin{aligned}
 p(u,w) = & a_{33} u^3 w^3 + a_{32} u^3 w^2 + a_{31} u^3 w + a_{30} u^3 + \\
 & a_{23} u^2 w^3 + a_{22} u^2 w^2 + a_{21} u^2 w + a_{20} u^2 + \\
 & a_{13} u w^3 + a_{12} u w^2 + a_{11} u w + a_{10} u + \\
 & a_{03} w^3 + a_{02} w^2 + a_{01} w + a_{00}
 \end{aligned} \quad (3.44)$$

Desde que cada um dos coeficientes vetoriais de a tem 3 componentes independentes, existem um total de 48 coeficientes ou 48 graus de liberdade.

A forma algébrica em notação matricial é dada por:

$$\mathbf{p} = \mathbf{UAW} \quad (3.45)$$

$$\text{onde: } \mathbf{U} = \begin{vmatrix} u^3 & u^2 & u & 1 \end{vmatrix} \quad (3.46)$$

$$\mathbf{W} = \begin{vmatrix} w^3 & w^2 & w & 1 \end{vmatrix} \quad (3.47)$$

$$\mathbf{A} = \begin{vmatrix} a_{33} & a_{32} & a_{31} & a_{30} \\ a_{23} & a_{22} & a_{21} & a_{20} \\ a_{13} & a_{12} & a_{11} & a_{10} \\ a_{03} & a_{02} & a_{01} & a_{00} \end{vmatrix} \quad (3.48)$$

Como visto com curvas, os coeficientes algébricos do retalho determinam a forma e posição no espaço da superfície. Apesar disto, retalhos de igual forma e tamanho tem um conjunto diferente de coeficientes se eles ocupam diferentes posições no espaço.

Um ponto no retalho é gerado para cada par específico de u, w na Equação (3.45) e apesar de u e w serem restritos ao intervalo $[0,1]$, o domínio das variáveis no espaço objeto x, y, z e o dos coeficientes algébricos é irrestrito.

Um retalho consiste de um número infinito de pontos dados pelas coordenadas x, y, z e portanto de um número infinito de pares u, w .

Assim na figura (Fig. 3.15) a seguir, se pode ver um retalho bicúbico mapeado no espaço objeto a partir de seus componentes no espaço paramétrico.

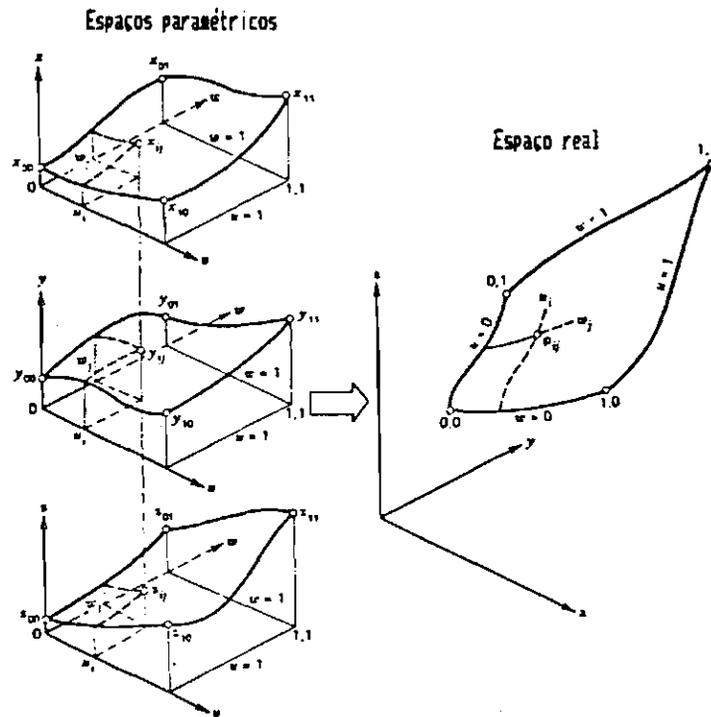


Fig. 3.15 - Superfície bicúbica mapeada no espaço objeto a partir de seus componentes no espaço paramétrico.

FONTE: Mortenson (1985), p. 158.

Observe que este retalho bicúbico é limitado por 4 curvas de contorno, onde cada curva é uma curva pc. As curvas são: u_0, u_1, w_0 e w_1 (para $u=0,1$ e $w=0,1$).

Outra maneira de ressaltar as curvas de contorno é: p_{0w}, p_{1w}, p_{u0} e p_{u1} , e para os pontos de esquina: p_{00}, p_{10}, p_{01} e p_{11} .

A figura (Fig. 3.16) a seguir, mostra a nomenclatura de um retalho bicúbico:

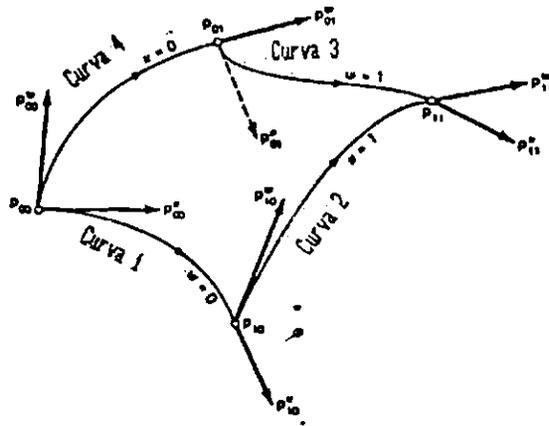


Fig. 3.16 - Nomenclatura para uma superfície bicúbica.
 FONTE: Mortenson (1985), p. 159.

Deriva-se a forma geométrica para a representação de um retalho bicúbico a partir da relação entre algumas condições de contorno e a forma algébrica.

Os 4 pontos de esquina do retalho: p_{00} , p_{10} , p_{01} e p_{11} , e os 8 vetores tangentes: p_{00}^u , p_{00}^w , p_{10}^u , p_{10}^w , p_{01}^u , p_{01}^w , p_{11}^u e p_{11}^w , são as condições de contorno que definem as curvas de contorno.

Além disso, precisa-se definir mais 4 vetores nos pontos de esquina que são as derivadas parciais mistas da função $p(u,w)$ e são chamados vetores de torsão ("twist"): p_{00}^{uw} , p_{10}^{uw} , p_{01}^{uw} e p_{11}^{uw} que são achados impondo as condições $u=0,1$ e $w=0,1$ em

$$d^2p(u,w)/dudw = 9a_{33}u^2w^2 + 6a_{32}u^2w + 3a_{31}u^2 + 6a_{23}uw^2 + 4a_{22}uw + 2a_{21}u + 3a_{13}w^2 + 2a_{12}w + a_{11} \quad (3.49)$$

Assim tem-se 16 vetores que proporcionam os 48 vetores geométricos.

A forma geométrica na forma matricial para o retalho bicúbico é dada por:

$$p = p(u,w) = UBM^T W^T \quad (3.50)$$

onde: U,W são as mesmas da Equação (3.45)

M é a matriz de transformação universal

$$B = \begin{vmatrix} p_{00} & p_{01} & p_{00}^w & p_{01}^w \\ p_{10} & p_{11} & p_{10}^w & p_{11}^w \\ p_{00}^u & p_{01}^u & p_{00}^{uw} & p_{01}^{uw} \\ p_{10}^u & p_{11}^u & p_{10}^{uw} & p_{11}^{uw} \end{vmatrix} \quad (3.51)$$

3.3.3 - ESPAÇO PARAMÉTRICO DE UMA SUPERFÍCIE

O espaço paramétrico de uma superfície de retalho difere do da curva desde que deve-se considerar uma variável paramétrica adicional. Portanto, tem-se um conjunto de espaços paramétricos tridimensionais definidos pelas coordenadas (u,w,x), (u,w,y), (u,w,z).

Assim como as curvas pc, decompor um retalho bicúbico em seus componentes paramétricos espaciais ajuda a entender seu comportamento no espaço objeto.

A figura (Fig. 3.17) a seguir, ilustra o espaço paramétrico:

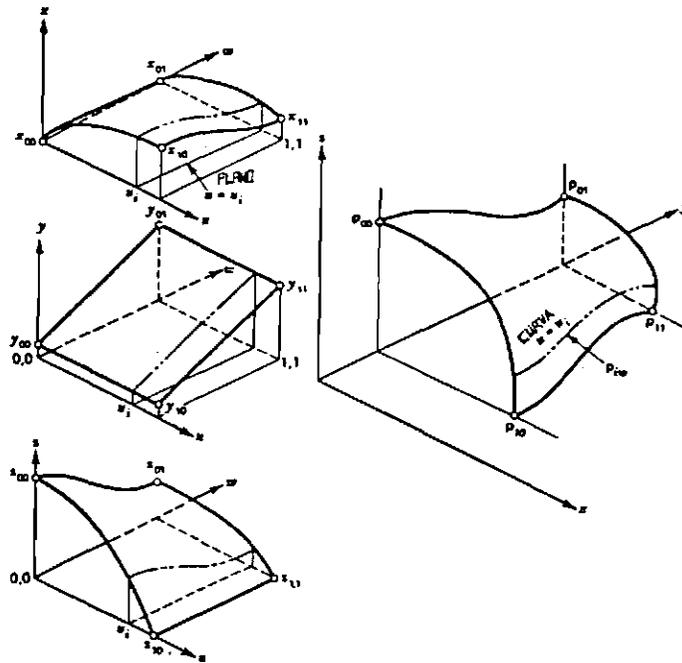


Fig. 3.17 - Espaço paramétrico de uma superfície.
 FONTE: Mortenson (1985), p. 312.

Na figura observa-se:

1. p_{00} e p_{01} ficam no plano yz e p_{10} e p_{11} ficam no plano xy . Cada curva de contorno é plana. Os gráficos à esquerda mostram os componentes do retalho no espaço paramétrico dados pelas equações $x=x(u,w)$, $y=y(u,w)$ e $z=z(u,w)$ respectivamente.
2. u e w estão restritas ao domínio $[0,1]$.

3. Mantendo-se uma das variáveis paramétricas constantes, define-se uma curva paramétrica específica no retalho. Aqui vê-se a curva p_{iw} . Todo ponto nesta curva tem o mesmo valor de u e no espaço paramétrico vemos esta curva decomposta em seus componentes, simplesmente passando o plano $u=u_i$ através de cada retalho componente.
4. Aqui, todos os retalhos tem forma algébrica idêntica, o que distingue um retalho do outro são seus coeficientes, pois são os coeficientes que controlam a posição do retalho no espaço assim como seu tamanho e forma.
5. Cada conjunto de valores u,w define um único ponto no espaço xyz no retalho bicúbico. Cada variável dependente x,y e z é controlada independentemente.

3.3.4 - SUPERFÍCIE PLANA

A figura (Fig. 3.18) a seguir, mostra a construção mais fácil a qual é interpretada como:

$$p(u,w) = p_{00} + ur + ws \quad u,w \in [0,1] \quad (3.52)$$

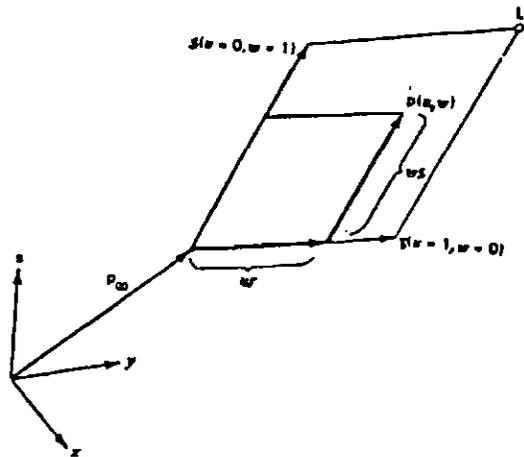


Fig. 3.18 - Equação vetorial de um plano.
 FONTE: Mortenson (1985), p. 185.

Esta equação define um retalho plano através do ponto p_{00} e paralelo aos vetores r e s . É um caso especial do retalho bicúbico da Equação (3.43) onde todos os coeficientes algébricos são zerados exceto a_{00} , a_{10} e a_{01} resultando:

$$p(u, w) = a_{00} + ua_{10} + wa_{01} \quad (3.53)$$

Esta equação é idêntica à Equação (3.52) onde acha-se $a_{00} = p_{00}$, $a_{10} = r$ e $a_{01} = s$.

Aplicando estes resultados determina-se a matriz B dos coeficientes geométricos como:

$$B = \begin{vmatrix} p_{00} & p_{00} + s & s & s \\ p_{00} + r & p_{00} + r + s & s & s \\ r & r & 0 & 0 \\ r & r & 0 & 0 \end{vmatrix} \quad (3.54)$$

3.3.5 - SUPERFÍCIE CILÍNDRICA

A superfície cilíndrica é um caso especial da superfície reguada ("ruled surface"). O cilindro é uma superfície gerada por uma linha reta conforme se move paralela a si mesma ao longo de uma curva, gerando um retalho bicúbico de superfície cilíndrica.

Na figura (Fig. 3.19) a seguir, pode se ver uma curva cujos coeficientes geométricos são

$$\begin{vmatrix} p_0 & p_1 & p_0^u & p_1^u \end{vmatrix}^T, \quad (3.55)$$

e um linha reta definida desde p_0 até p_2 .

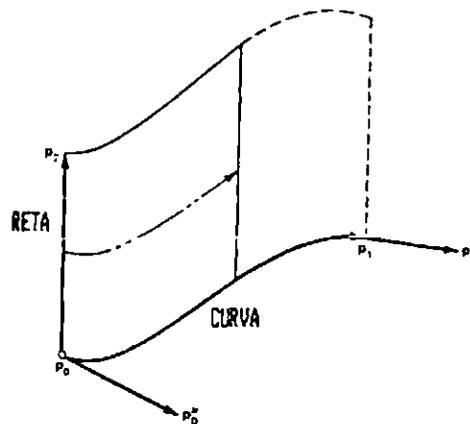


Fig. 3.19 - Superfície cilíndrica.

FONTE: Mortenson (1985), p. 189.

Usando estes parâmetros calcula-se a matriz B para o retalho definido pela curva e a reta, e é dada por:

$$B = \begin{vmatrix} p_0 & p_2 & p_2 - p_0 & p_2 - p_0 \\ p_1 & p_1 + p_2 - p_0 & p_2 - p_0 & p_2 - p_0 \\ p_1^u & p_1^u & 0 & 0 \\ p_1^0 & p_1^0 & 0 & 0 \end{vmatrix} \quad (3.56)$$

3.3.6 - SUPERFÍCIE REGUADA

A trajetória de uma linha reta movendo-se com 1 grau de liberdade determina uma superfície. Este tipo de superfície é chamada superfície reguada.

Pode-se definir uma superfície reguada como uma superfície tal que através de cada ponto dela passe pelo menos uma linha reta que esteja totalmente dentro da superfície.

O plano é o caso mais simples. O cone, o cilindro, o parabolóide hiperbólico são casos especiais.

Toda superfície desenrolável (que pode ser desenrolada em um plano sem contrair-se ou distorcer-se) é reguada, porém toda superfície reguada não é desenrolável.

A figura (Fig. 3.20) a seguir, ilustra os componentes do retalho de superfície reguada bicúbica.

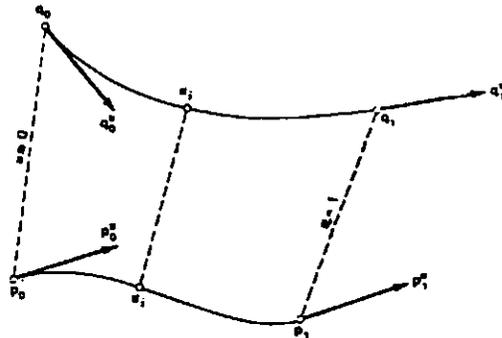


Fig. 3.20 - Superfície reguada.

FONTE: Mortenson (1985), p. 192.

Dadas duas curvas $p(u)$ e $q(u)$ constroi-se uma superfície reguada unindo com uma linha reta cada ponto em $p(u)$ a um ponto em $q(u)$ que tenha equivalente valor de u . A matriz B é dada por:

$$B = \begin{vmatrix} p_0 & q_0 & q_0 - p_0 & q_0 - p_0 \\ p_1 & q_1 & q_1 - p_1 & q_1 - p_1 \\ p_0^u & q_0^u & 0 & 0 \\ p_1^u & q_1^u & 0 & 0 \end{vmatrix} \quad (3.57)$$

3.3.7 - SUPERFÍCIE DE REVOLUÇÃO

Este tipo de superfície é semelhante à superfície do sólido primitivo usado na implementação pois este também é gerado por revolução.

Pode-se gerar uma superfície de revolução rotacionando uma curva plana em torno a um eixo coplanar a ela. Observe a figura (Fig. 3.21) a seguir:

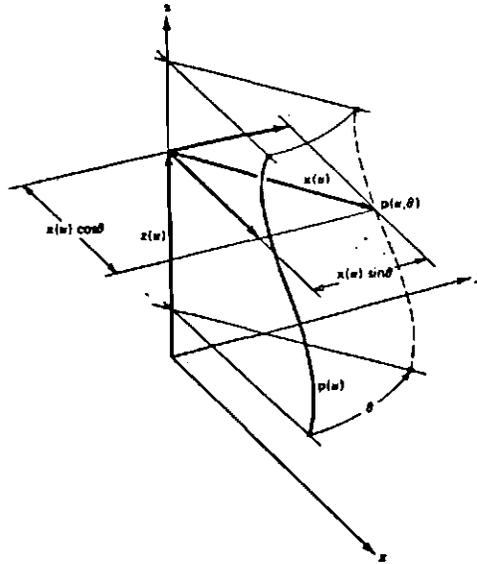


Fig. 3.21 - Superfície de revolução.

FONTE: Mortenson (1985), p. 195.

A curva plana é chamada curva de perfil ou de contorno, a qual nas suas várias posições no espaço cria meridianos. Os círculos criados por cada ponto desta curva são chamados paralelos.

Para o caso mais simples, seja o eixo z, o eixo de rotação, e seja a curva $p(u)=x(u)+z(u)$ definida no plano xz, então a superfície de revolução tem a equação da forma:

$$p(u,0) = x(u) \cos\theta + x(u) \sin\theta + z(u) \quad (3.58)$$

A curva de perfil pode ser uma curva geral, um segmento elíptico, arco circular, etc.

Para uma superfície de revolução mais geral, considere a situação vista na figura (Fig. 3.22) a seguir:

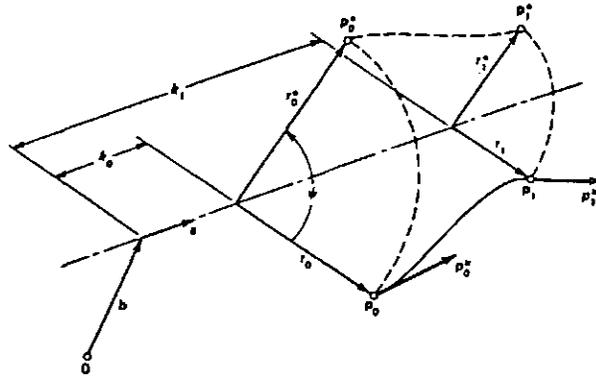


Fig. 3.22 - Análise de uma superfície de revolução.
 FONTE: Mortenson (1985), p. 196.

Os elementos dados são uma curva pc definida por seus coeficientes geométricos, um vetor unitário \mathbf{a} definindo a direção do eixo de revolução, um vetor \mathbf{b} localizando um ponto através do qual o eixo passa e o ângulo θ através do qual a curva é rotacionada para varrer e definir a superfície.

A idéia é achar um retalho bicúbico que descreva esta superfície em função destes elementos: \mathbf{a} , \mathbf{b} , θ , \mathbf{p}_0 , \mathbf{p}_0^u , \mathbf{p}_1 e \mathbf{p}_1^u .

Primeiro devem ser determinados os escalares k_0 e k_1 e os vetores \mathbf{r}_0 e \mathbf{r}_1 . Isto pode ser conseguido resolvendo as seguintes equações vetoriais:

$$\mathbf{b} + k_0 \mathbf{a} + \mathbf{r}_0 = \mathbf{p}_0 \quad (3.59)$$

$$\mathbf{a} \cdot \mathbf{r}_0 = 0 \quad (3.60)$$

$$\mathbf{b} + k_1 \mathbf{a} + \mathbf{r}_1 = \mathbf{p}_1 \quad (3.61)$$

$$\mathbf{a} \cdot \mathbf{r}_1 = 0 \quad (3.62)$$

\mathbf{a} e \mathbf{r}_0 , assim como \mathbf{a} e \mathbf{r}_1 são mutuamente perpendiculares, logo das Equações (3.59) e (3.60) são obtidas:

$$k_0 = \mathbf{a} \cdot (\mathbf{p}_0 - \mathbf{b}) \quad (3.63)$$

$$\mathbf{r}_0 = \mathbf{p}_0 - k_0 \mathbf{a} - \mathbf{b} \quad (3.64)$$

das Equações (3.61) e (3.62) são obtidas:

$$k_1 = \mathbf{a} \cdot (\mathbf{p}_1 - \mathbf{b}) \quad (3.65)$$

$$\mathbf{r}_1 = \mathbf{p}_1 - k_1 \mathbf{a} - \mathbf{b} , \quad (3.66)$$

agora podem ser achados os outros dois pontos de esquina \mathbf{p}_0^* e \mathbf{p}_1^* :

$$\mathbf{p}_0^* = \mathbf{b} + k_0 \mathbf{a} + \mathbf{r}_0^* \quad (3.67)$$

$$\mathbf{p}_1^* = \mathbf{b} + k_1 \mathbf{a} + \mathbf{r}_1^* \quad (3.68)$$

finalmente, basta determinar \mathbf{r}_0^* e \mathbf{r}_1^* os quais são determinados da relação das tangentes unitárias com o raio e o ângulo de rotação.

Assim consegue-se determinar os 4 vetores de esquina que definem o retalho de superfície e a partir destes valores calculam-se os 8 vetores tangentes e os 4 vetores de torsão nas esquinas do retalho, e portanto montar a matriz dos coeficientes geométricos.

Não é apresentado o desenvolvimento das equações por ser muito extenso, porém a ideia básica foi dada.

3.3.8 - SUPERFÍCIE ESFÉRICA

Este tipo de superfície foi obtida pela varredura de um semicírculo (contorno fechado composto de uma reta e um arco) em torno de um eixo de rotação.

Na figura (Fig. 3.23) a seguir, está ilustrada uma superfície esférica centrada na origem, de raio r ; θ , o ângulo formado pelo vetor radial, que aponta o ponto $p(\theta, \varphi)$, com o eixo z ; e ângulo φ , que é tomado em torno do eixo z e medido em relação ao eixo x :

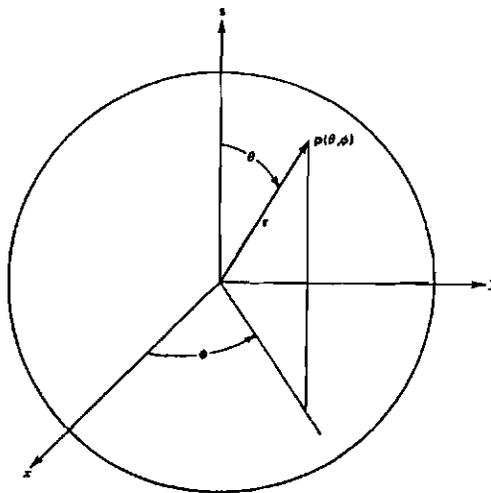


Fig. 3.23 - Superfície esférica.

FONTE: Mortenson (1985), p. 200.

A equação que descreve a esfera, é dada por:

$$\mathbf{p}(\theta, \varphi) = (r \sin\theta \cos\varphi)\mathbf{i} + (r \sin\theta \sin\varphi)\mathbf{j} + (r \cos\theta)\mathbf{k} \quad (3.69)$$

onde i, j, k são os vetores unitários nas direcções x, y, z respectivamente.

3.3.9 - SUPERFÍCIE DE BÉZIER

Assim como a curva de Bézier tem um polígono característico, a superfície de Bézier tem um poliedro característico.

Os pontos na superfície de Bézier são dados por uma simples extensão da equação geral para os pontos em uma curva Bézier (Equação 3.30):

$$p(u, w) = \sum_{i=0}^m \sum_{j=0}^n p_{i,j} B_{i,n}(u) B_{j,n}(w) \quad (3.70)$$

para: $u, w \in [0, 1]$,

onde $p_{i,j}$ são os vértices do polinômio característico que formam uma matriz retangular $(m+1) \times (n+1)$ de pontos, e $B_{i,n}$ e $B_{j,n}$ são definidos tal como para as curvas usando as Equações (3.29) e (3.30).

A superfície de Bézier não precisa necessariamente ser representada por uma matriz quadrada de pontos de controle. Expressa-se um retalho bicúbico de Bézier similarmente ao retalho bicúbico de Hermite.

Usando a representação binomial da curva cúbica de Bézier, a equação matricial para um retalho definido por uma matriz 4×4 de pontos é:

$$p(u,w) = |(1-u)^3 \quad 3u(1-u)^2 \quad 3u^2(1-u) \quad u^3| P K \quad (3.71)$$

onde:

$$K = \begin{vmatrix} (1-w)^3 \\ 3w(1-w)^2 \\ 3w^2(1-w) \\ w^3 \end{vmatrix} \quad P = \begin{vmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{vmatrix} \quad (3.72)$$

A matriz P contém os vetores de posição para pontos que definem o poliedro característico e o portanto o retalho de superfície de Bézier. Na figura (Fig. 3.24) a seguir, são ilustrados estes pontos, o poliedro e o retalho resultante:

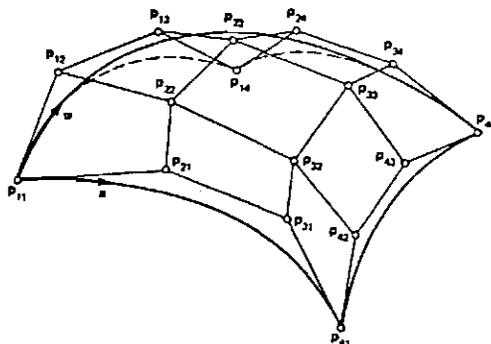


Fig. 3.24 - Superfície cúbica de Bézier.

FONTE: Mortenson (1985), p. 216.

Aqui, somente os 4 pontos de esquina p_{11} , p_{41} , p_{14} e p_{44} descansam na superfície. Os pontos p_{21} , p_{31} , p_{12} , p_{13} , p_{42} , p_{43} , p_{24} e p_{34} controlam as inclinações das curvas de controle.

Os 4 pontos restantes p_{22} , p_{32} , p_{23} e p_{33} controlam as inclinações transversais ("cross slopes") ao longo das curvas de contorno da mesma maneira que os vetores de torção do retalho bicúbico.

Tal como mostra a figura, a superfície de Bézier é definida completamente por uma malha de pontos descrevendo duas famílias de curvas de Bézier na superfície. Cada curva é definida por um polígono de 4 vértices. Pode ser usado um número maior de pontos, resultando em um polinômio de maior grau.

3.3.10 - SUPERFÍCIE B-SPLINE

Este tipo de superfície faz parte da superfície de um sólido primitivo na implantação, especificamente daquele que é gerado pela varredura de um contorno fechado onde um ou vários elementos deste contorno são curvas B-Spline. Assim, a seguir é focado este tipo de superfície.

A formulação da superfície B-Spline baseia-se nas curvas B-Spline, semelhante à relação das curvas e superfícies de Bézier.

Analogamente, a superfície B-Spline é definida em termos de um poliedro característico. A forma da superfície se aproxima do polígono. A aproximação é maior conforme os valores de k e l crescem. Veja a equação geral:

$$p(u,w) = \sum_{i=0}^m \sum_{j=0}^n p_{ij} N_{i,k}(u) N_{j,l}(w) \quad (3.73)$$

Os p_{ij} são os vértices do poliedro definido e os $N_{i,k}(u)$ e $N_{j,l}(w)$ são as funções de blending da mesma forma que aquelas das curvas B-Spline na Equação (3.35).

O grau de cada um dos polinômios de funções de blending $N_{i,k}(u)$ e $N_{j,l}(w)$ é controlado por k e l respectivamente e são computados recursivamente usando as Equações (3.36) à (3.38).

A forma matricial para a superfície B-Spline é semelhante a forma matricial para a curva B-Spline. Assim como a curva B-Spline é computada em segmentos de intervalos unitários na variação da variável paramétrica u , um quadrado unitário nas variáveis paramétricas u e w é usado para computar retalhos na superfície B-Spline.

A forma matricial geral de uma superfície B-Spline periódica aberta que aproxima uma matriz ("array") retangular de pontos $(m+1) \times (n+1)$ é dada por:

$$p_{st}(u,w) = U_k^M P_{kl} M_l^T W_l^T \quad (3.74)$$

onde: $s \in |1, m+2-k|$
 $t \in |1, n+2-l|$
 $u, w \in |0, 1|$

k e l denotam os parâmetros que controlam a continuidade da superfície e o grau do polinômio das funções blending, s e t identificam um retalho particular na superfície.

$$U_k = | u^{k-1} \quad u^{k-2} \quad \dots \quad u \quad 1 | \quad (3.75)$$

$$W_l = | w^{l-1} \quad w^{l-2} \quad \dots \quad w \quad 1 | \quad (3.76)$$

Os elementos da matriz $k \times l$ de pontos de controle: p_{kl} , dependem do retalho particular a ser avaliado.

As matrizes M_k e M_l são idênticas à matriz de transformação para curvas B Spline, as quais seguem um desenvolvimento mais complexo e não foram abordadas aqui.

A figura (Fig. 3.25) a seguir, mostra dois retalhos de superfície B-Spline para $k=3$:

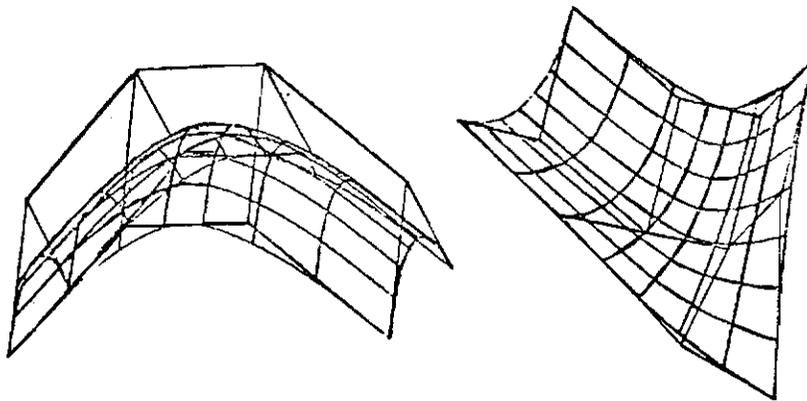


Fig. 3.25 - Superfícies B-Spline.

FONTE: Newman e Sproull (1979),
p. 326.

3.4 - SÓLIDOS

A biblioteca gráfica da estação de trabalho inclui sub-rotinas que permitem gerar objetos tridimensionais como sólidos reais.

São sólidos reais porque o sistema CAD consegue representar o interior do sólido assim como propriedades internas e comportamento interno.

Neste subcapítulo é apresentado o suporte matemático e a geometria para uma forma de representação completa de um sólido. Neste suporte, está incluído o conceito de continuidade entre as faces dos sólidos que se unem ou se interceptam, dando uma idéia de operações booleanas.

Assim como para curvas e superfícies, os sólidos também são representados matricialmente com suas formas algébricas e geométricas.

3.4.1 - DEFINIÇÕES

O elemento matemático mais simples e direto para modelar um sólido é o sólido paramétrico ou HIPERPATCH. O sólido paramétrico é uma coleção de pontos de retalhos ("patch") limitados cujas coordenadas são dadas por funções contínuas e unívocas, dados três parâmetros u, v e w da forma:

$$x = x(u,v,w) \quad y = y(u,v,w) \quad z = z(u,v,w) \quad (3.77)$$

Onde as variáveis paramétricas u, v, w são limitadas ao intervalo $[0,1]$.

Fixando o valor de uma das variáveis paramétricas obtém-se uma superfície dentro ou no contorno do sólido paramétrico em termos das outras duas variáveis que ficam livres.

Continuando com este processo para qualquer número de valores arbitrários no intervalo permitido, primeiro para uma variável e logo para cada uma das outras, forma-se uma malha ou grade paramétrica de células de 3 famílias biparamétricas de superfície através de todo o sólido.

Através de cada ponto $p(u,v,w)$ passa apenas uma superfície de cada família. Estas superfícies também são chamadas superfícies isoparamétricas indicando uma superfície dentro do sólido, na qual uma das 3 variáveis paramétricas é constante.

Associado com cada sólido paramétrico existe um conjunto de elementos de contorno, como mostra a figura (Fig. 3.26) a seguir:

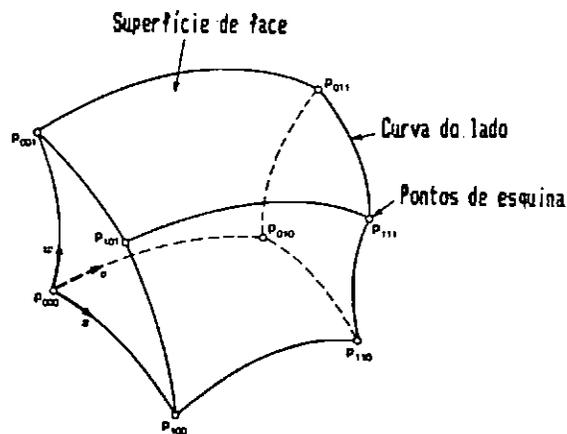


Fig. 3.26 - Elementos de contorno de um sólido paramétrico.
 FONTE: Mortenson (1985), p. 241.

O conjunto mais óbvio, é constituído de 8 pontos de esquina, 12 curvas definindo os lados e 6 superfícies (retalhos) definindo as faces. Outros incluem os vetores tangentes e vetores de torsão nas curvas dos lados e nos retalhos de face.

Para um sólido paramétrico ordinário não degenerado, existem sempre 8 pontos de esquina. Isto é consequência da possível combinação dos dois limites das 3 variáveis paramétricas.

Então os pontos de esquina são achados substituindo as 8 combinações de 0 e 1 em $p(u,v,w)$ para obter: $p(0,0,0)$, $p(1,0,0)$, $p(0,1,0)$, $p(0,0,1)$, $p(1,1,0)$, $p(1,0,1)$, $p(0,1,1)$ e $p(1,1,1)$.

As curvas dos lados (arestas) são funções de uma das 3 variáveis paramétricas. Elas são obtidas, permitindo que uma das variáveis fique livre enquanto combinações fixas sucessivas das outras 2 variáveis nos seus valores limites são processadas.

Isto dá como resultado 12 possíveis combinações e portanto a definição de funções das 12 curvas paramétricas das arestas: $p(u,0,0)$, $p(u,1,0)$, $p(u,0,1)$, $p(u,1,1)$, $p(0,v,0)$, $p(1,v,0)$, $p(0,v,1)$, $p(1,v,1)$, $p(0,0,w)$, $p(1,0,w)$, $p(0,1,w)$ e $p(1,1,w)$.

As faces dos retalhos são funções de duas das 3 variáveis paramétricas. Elas são obtidas, deixando livres duas das variáveis enquanto fixamos sucessivamente a variável restante à cada um dos seus valores limites. Isto dá como resultado 6 funções de "patches" possíveis: $p(u,v,0)$, $p(u,v,1)$, $p(u,0,w)$, $p(u,1,w)$, $p(0,v,w)$ e $p(1,v,w)$.

O exemplo mais simples é o sólido retangular mostrado na figura (Fig. 3.27) a seguir:

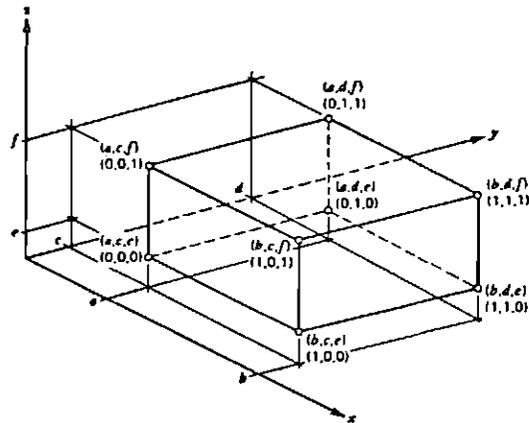


Fig. 3.27 - Sólido retangular paramétrico.
 FONTE: Mortenson (1985), p. 242.

As equações paramétricas abaixo representam o sólido em um sistema de coordenadas xyz dado por:

$$x = (b-a)u+a \quad y = (d-c)v+c \quad z = (f-e)w+e \quad (3.78)$$

onde: $u, v, w \in [0, 1]$.

Estas equações definem não somente os pontos que limitam o sólido como também seus pontos interiores.

3.4.2 - FORMA ALGÉBRICA E GEOMÉTRICA

Analogamente à forma cúbica para curvas, bicúbica para superfícies, o sólido é apresentado na forma tricúbica.

A forma algébrica para um sólido tricúbico tem a seguinte equação polinomial:

$$p(u,v,w) = \sum_{i=0}^3 \sum_{j=0}^3 \sum_{k=0}^3 a_{ijk} u^i v^j w^k \quad (3.79)$$

$$u,v,w \in [0,1] ,$$

a_{ijk} são os coeficientes algébricos do sólido. O termo tricúbico deve-se ao fato de que cada uma das 3 variáveis paramétricas pode aparecer como um termo cúbico.

A restrição das variáveis paramétricas u,v,w ao intervalo $[0,1]$ inclusive, torna o sólido a ser limitado de uma maneira regular por retalhos cúbicos.

Desenvolvendo o polinômio cúbico em u,v,w para $x(u,v,w)$ obtém-se:

$$x(u,v,w) = a_{333} u^3 v^3 w^3 + a_{332} u^3 v^3 w^2 + \dots + a_{000} x \quad (3.80)$$

analogamente, conseguem-se expressões similares para $y(u,v,w)$ e $z(u,v,w)$.

Usando notação vetorial para (3.79) vem:

$$\begin{aligned}
p(u,v,w) = & a_{333}u^3v^3w^3 + a_{332}u^3v^3w^2 + a_{331}u^3v^3w + a_{330}u^3v^3 \\
& + a_{323}u^3v^2w^3 + \dots \\
& : \\
& + a_{000}
\end{aligned} \tag{3.81}$$

Expresando matricialmente esta equação obtém-se:

$$p(u,v,w) = UA_{uv}V^T A_w W^T \quad u,v,w \in [0,1] \tag{3.82}$$

$$\begin{aligned}
\text{onde: } U &= \begin{vmatrix} u^3 & u^2 & u & 1 \end{vmatrix} \\
V^T &= \begin{vmatrix} v^3 & v^2 & v & 1 \end{vmatrix}^T \\
W^T &= \begin{vmatrix} w^3 & w^2 & w & 1 \end{vmatrix}^T
\end{aligned} \tag{3.83}$$

A_{uv} é o produto do primeiro e segundo polinómios cúbicos em u e v , A_w é o terceiro polinómio em w .

Com outra escolha de indexação, se tem:

$$p(u,v,w) = \sum_{i=1}^4 \sum_{j=1}^4 \sum_{k=1}^4 a_{ijk} u^{4-i} v^{4-j} w^{4-k} \tag{3.84}$$

e pode-se simplificar eliminando os somatórios e adotando a convenção de que para cubos, a variação dos índices é de 1 á 4, tal que:

$$p = a_{ijk} u^{4-i} v^{4-j} w^{4-k} \tag{3.85}$$

Pode ser usado o mesmo recurso para a forma geométrica obtendo:

$$p = F_i(u)F_j(v)F_k(w)b_{ijk} \tag{3.86}$$

os termos F são as funções blending conhecidas F_1, F_2, F_3 e F_4 em termos de valores específicos das variáveis paramétricas. b_{ijk} é o arranjo de condições de contorno ou vetores geométricos.

Em cada uma das 8 esquinas temos as seguintes condições de contorno: o próprio ponto de esquina, três vetores tangentes, três vetores de torsão e um vetor definido pela derivada parcial mista de ordem 3 da função.

Por exemplo, olhando a figura (Fig. 3.28) a seguir, de um cubo unitário no espaço paramétrico e escolhendo-se a esquina definida por $p(1,0,1)$, as condições de contorno são:

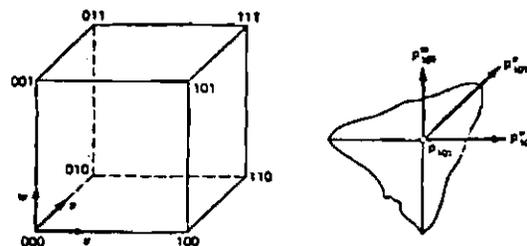


Fig. 3.28 - Coeficientes geométricos do cubo unitário.

FONTE: Mortenson (1985), p. 245.

ponto de esquina: $p(1,0,1)$, vetores tangentes $d/du, d/dv, d/dw$: $p_{101}^u, p_{101}^v, p_{101}^w$, vetores de torsão $d^2/dudv, d^2/dudw, d^2/dvdw$: $p_{101}^{uv}, p_{101}^{uw}, p_{101}^{vw}$ e deriva-da parcial mista $d^3/dudvdw$: p_{101}^{uvw} .

A interpretação do vetor b_{ijk} leva a uma matriz $4 \times 4 \times 4 \times 3$ quadrimensional de vetores geométricos.

3.4.3 - VETORES TANGENTES E VETORES DE TORSÃO

A função tricúbica é expressa por 3 variáveis independentes, por isso é necessário calcular derivadas parciais desta função.

Tem-se 3 tipos de vetores tangentes: \mathbf{p}_{uvw}^u , \mathbf{p}_{uvw}^v , \mathbf{p}_{uvw}^w , 3 vetores de torsão: \mathbf{p}_{uvw}^{uv} , \mathbf{p}_{uvw}^{uw} e \mathbf{p}_{uvw}^{vw} e uma derivada parcial mista: \mathbf{p}_{uvw}^{uvw} .

A figura (Fig. 3.29) a seguir, ilustra estes vetores:

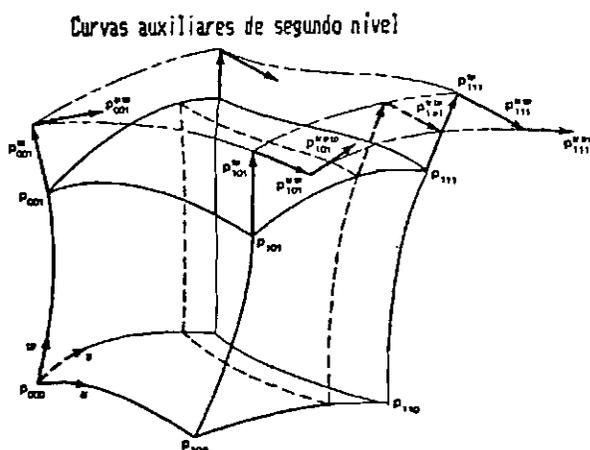


Fig. 3.29 - Vetores tangentes do sólido paramétrico.

FONTE: Mortenson (1985), p. 252.

Conforme estes vetores mudam, controlam a forma das curvas e superfícies e portanto controlam a forma do sólido tricúbico. Além de controlar a forma dos lados e faces, também controlam a distribuição interna das variáveis paramétricas.

3.4.4 - ESPAÇO PARAMÉTRICO DO SÓLIDO

O espaço paramétrico de um sólido tem uma variável a mais do que o espaço paramétrico da curva ou superfície. Consiste de um conjunto de 4 espaços paramétricos quadridimensionais definidos pelas coordenadas (u,v,w,x) , (u,v,w,y) e (u,v,w,z) .

Ao contrário do que para curvas e superfícies, onde os gráficos dos seus componentes no espaço paramétrico ajudam a visualizar seu comportamento no espaço objeto, decompor um sólido tricúbico nos seus componentes de espaço paramétrico dificulta a visualização e entendimento do seu comportamento no espaço objeto.

Porém, pode-se aplicar esta técnica às faces do sólido paramétrico da mesma maneira que para os retalhos bicúbicos. Assim pode-se tomar fatias paramétricas do sólido gerando superfícies nas quais uma das variáveis é constante e logo proceder com a decomposição bicúbica.

A figura (Fig. 3.30) a seguir, mostra esta técnica aplicada a um cubo unitário.

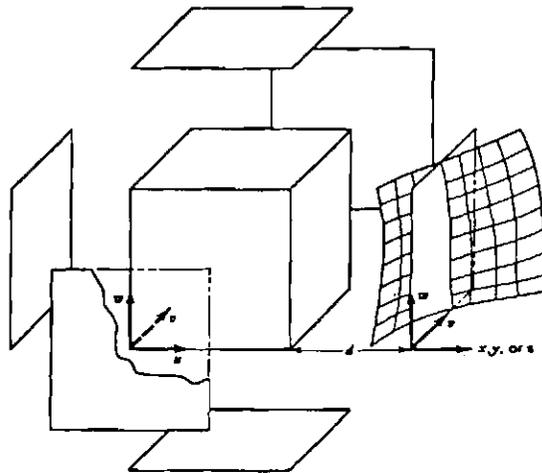


Fig. 3.30 - Espaço paramétrico de um sólido paramétrico.
 FONTE: Mortenson (1985), p. 253.

3.4.5 - CONTINUIDADE E SÓLIDOS COMPOSTOS

Isto ilustra o conceito matemático de união booleana para compor um sólido a partir de dois sólidos mais simples.

Considere a continuidade de dois sólidos unidos $p(u,v,w)$ e $q(u,v,w)$ na superfície comum de fronteira dando como resultado um sólido composto como mostra a figura (Fig. 3.31) a seguir:

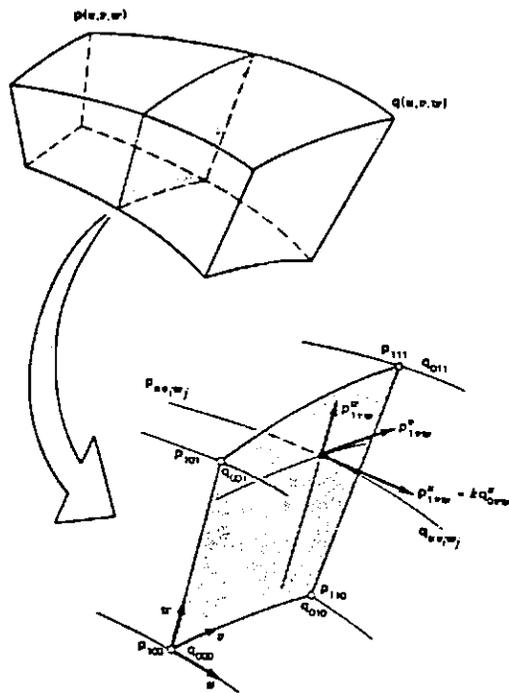


Fig. 3.31 - Condições de continuidade.

FONTE: Mortenson (1985), p. 254.

Para assegurar continuidade C^0 (que significa que não devem existir buracos ou pontos de quebra nas superfícies de contato) deve-se ter a seguinte condição na superfície comum:

$$p_{1vw} = q_{0vw} \quad (3.87)$$

Para continuidade C^1 (que significa que ambas tangentes devem ter a mesma direção no ponto em comum de contato), os vetores tangentes nos pontos em comum das curvas de v e w constantes, na figura (Fig. 3.31) anterior, devem ser colineares e atender à:

$$p_{1vw}^u = k q_{Ovw}^u \quad e \quad p_{1vw}^v \times p_{1vw}^w = b p_{1vw}^u \quad (3.88)$$

onde: b e k são constantes.

3.5 - TRANSFORMAÇÕES

O modelo geométrico de um objeto deve ser fácil de manipular ou de transformar. Transformar um objeto implica na alteração da posição, orientação e de forma.

Desde que o modelo geométrico deve ser guardado no computador, como dados numéricos em termos de algum sistema de coordenadas, a maneira mais fácil de transformar os dados, para representar mudança na posição e orientação de um objeto, é através de operações entre matrizes.

As transformações se dividem em 3 tipos:

- Transformações de coordenadas ou transformações de modelamento ("modeling transformations"), para transladar, mudar de escala ou rotacionar um objeto;
- Transformações de visualização ("viewing transformations"), para mudar o ponto de vista em relação ao objeto;
- Transformações de projeção ("projection transformations"), para determinar como o objeto deve ser visto.

Existem 4 sistemas de coordenadas usadas para executar transformações de coordenadas:

- Sistema de coordenadas do objeto ("object coordinate system"), que é o próprio sistema de coordenadas do objeto, por exemplo, o sistema de coordenadas de um cubo. É independente de qualquer outro sistema de coordenadas.
- Sistema de coordenadas do mundo ("world coordinate system"), que são as coordenadas do mundo no qual o objeto existe. As coordenadas de cada objeto são relativas às coordenadas dos outros objetos neste mundo.
- Sistema de coordenadas da tela ou superfície de exibição ("screen coordinates system"), o qual é o sistema de coordenadas no qual o objeto transformado aparece na tela. Uma janela ("viewport") dentro da tela determina as coordenadas da tela.
- Sistema de coordenadas do olho do observador ("eye coordinate system"), o qual é o sistema de coordenadas no qual o olho se posiciona na origem e olha na direção negativa do eixo z.

Os 4 passos básicos usuais no processo de construção de um objeto estão relacionados com as transformações de coordenadas:

1. Constrói-se o objeto gráfico usando coordenadas.
2. Transforma-se o objeto gráfico a uma localização conveniente dentro do sistema de coordenadas do mundo. Usam-se as transformações de modelamento: translação, rotação ou escala.

3. Especifica-se a posição do olho do observador e a direção de visualização para colocar o olho em um ponto de interesse ao longo do eixo -z. Usam-se as transformações de visualização.
4. Escolhe-se uma projeção para especificar qual região do objeto gráfico aparecerá na tela. Usam-se transformações de projeção.

3.5.1 - TRANSFORMAÇÕES DE MODELAMENTO

Cada objeto gráfico, ou modelo geométrico, é definido no seu próprio sistema de coordenadas.

As transformações de modelamento são usadas para expressar a localização dos objetos em relação aos outros.

Um objeto pode ser movido de duas maneiras: transformando suas coordenadas em relação a seu sistema de coordenadas (onde é medida a posição do objeto), ou movendo seu sistema de coordenadas de referência.

O objeto inteiro pode ser manipulado usando as transformações de modelamento: translação, rotação e escala.

Cada transformação é expressada como uma entidade única, a matriz de transformação.

Transformações complexas, expressadas como uma sequência de transformações primitivas, podem ser concatenadas para produzir uma única matriz de transformação que tem o mesmo efeito que a sequência de transformações.

3.5.1.1 - TRANSLAÇÃO

A transformação que translada um ponto (x,y,z) do modelo geométrico para outro ponto (x',y',z') é dada por

$$|x' \ y' \ z' \ 1| = |x \ y \ z \ 1| \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{vmatrix} \quad (3.90)$$

$$\text{ou: } X_T = X_O \cdot T(t_x, t_y, t_z), \quad (3.91)$$

onde t_x , t_y e t_z são as translações nas direções x , y e z respectivamente. T representa a matriz de translação.

Para um objeto geométrico bidimensional, em um sistema de coordenadas 2D, a Equação (3.90) se reduz à:

$$|x \ y \ 1| = |x \ y \ 1| \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{vmatrix} \quad (3.92)$$

3.5.1.2 - ROTAÇÃO

Para rotacionar um ponto do modelo geométrico, inicialmente deve-se determinar um sistema de 3 eixos em torno do qual rotacionar.

Considerando os eixos de coordenadas mostrados na figura (Fig. 3.32) a seguir, e usando como convenção a regra da mão esquerda:

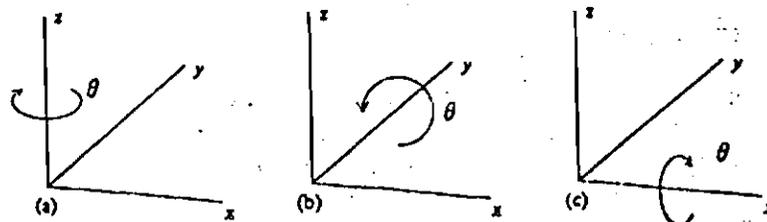


Fig. 3.32 - Convenção do sentido de rotação positivo.

FONTE: Newman e Sproull (1979), p. 234.

A rotação em torno da origem do eixo de coordenadas z de um ângulo θ é dada pela seguinte transformação:

$$\begin{vmatrix} x' & y' & z' & 1 \\ x & y & z & 1 \end{vmatrix} = \begin{vmatrix} \cos\theta & -\text{sen}\theta & 0 & 0 \\ \text{sen}\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (3.93)$$

$$\text{ou: } X_T = X_O \cdot R_Z(\theta) \quad , \quad (3.94)$$

analogamente, a rotação em torno da origem do eixo de coordenadas y de um ângulo θ é dada pela seguinte transformação:

$$|x' \ y' \ z' \ 1| = |x \ y \ z \ 1| \begin{vmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (3.95)$$

$$\text{ou: } X_T = X_o \cdot R_y(\theta), \quad (3.96)$$

analogamente, a rotação em torno da origem do eixo de coordenadas x de um ângulo θ é dada pela seguinte transformação:

$$|x' \ y' \ z' \ 1| = |x \ y \ z \ 1| \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (3.97)$$

$$\text{ou: } X_T = X_o \cdot R_x(\theta) \quad (3.98)$$

$R_x(\theta)$, $R_y(\theta)$ e $R_z(\theta)$, representam as matrizes de rotação em torno dos eixos x , y e z , respectivamente

Para um objeto geométrico em duas dimensões, em um sistema de coordenadas xy , a rotação em torno da origem de um ângulo θ é dada por:

$$|x' \ y' \ 1| = |x \ y \ 1| \begin{vmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad (3.99)$$

Para rotacionar o modelo geométrico em torno de um ponto arbitrário, devem-se concatenar 3 transformações: primeiro, transladar o modelo à origem, depois, rotacionar o modelo e finalmente, transladar o modelo de volta ao ponto de origem.

3.5.1.3 - MUDANÇA DE ESCALA

Uma transformação de escala pode ser usada para mudar a escala em cada uma das direções das coordenadas separadamente:

$$|x' \ y' \ z' \ 1| = |x \ y \ z \ 1| \begin{vmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (3.100)$$

$$\text{ou: } X_T = X_O \cdot S(s_x, s_y, s_z) \quad (3.101)$$

onde s_x , s_y e s_z são os fatores de escala nas direções x , y e z respectivamente. S representa a matriz de escala.

Para um objeto geométrico bidimensional, em um sistema de coordenadas 2D, a Equação (3.100) se reduz à:

$$|x' \ y' \ 1| = |x \ y \ 1| \begin{vmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad (3.102)$$

3.5.1.4 - TRANSFORMAÇÕES INVERSAS

Várias transformações dadas acima tem uma inversa, a qual executa um transformação simetricamente oposta.

Por exemplo, a inversa da Matriz (3.90) que cancela o efeito de translação desta matriz, é dada por:

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -T_x & -T_y & -T_z & 1 \end{vmatrix} \quad (3.103)$$

Outro exemplo é a inversa da matriz da Equação (3.93) que representa uma rotação do mesmo módulo, em torno do mesmo eixo, porém em sentido contrário, dada por:

$$\begin{vmatrix} \cos-\theta & -\text{sen}-\theta & 0 & 0 \\ \text{sin}-\theta & \cos-\theta & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (3.104)$$

3.5.1.5 - COMPOSIÇÃO

A aplicação sucessiva de um número de transformações pode ser obtida com uma única matriz de transformação, a concatenação da sequência.

Suponha que duas transformações T_1 e T_2 serão aplicadas sucessivamente no modelo. O mesmo efeito pode ser conseguido aplicando uma única transformação T_3 no modelo o qual é simplesmente o produto de T_1 por T_2 .

Isto pode ser demonstrado como segue:

o ponto (x,y,z) é transformado em (x',y',z') por T_1 , tal que:

$$|x' \ y' \ z' \ 1| = |x \ y \ z \ 1| T_1, \quad (3.105)$$

gera-se outro ponto (x'',y'',z'') aplicando T_2 em (x',y',z') , tal que:

$$|x'' \ y'' \ z'' \ 1| = |x' \ y' \ z' \ 1| T_2 \quad (3.106)$$

Substituindo a Equação (3.105) em (3.106), se tem:

$$\begin{aligned} |x'' \ y'' \ z'' \ 1| &= (|x \ y \ z \ 1| T_1) T_2 \\ &= |x \ y \ z \ 1| (T_1 \cdot T_2) \end{aligned} \quad (3.107)$$

ou seja, o ponto original (x,y,z) foi trasladado ao ponto final (x'',y'',z'') aplicando uma única transformação dada por $T_1 \cdot T_2$.

A ordem em que se concatena as matrizes de transformação deve ser preservada, pois este processo não é comutativo.

3.5.2 - TRANSFORMAÇÕES DE VISUALIZAÇÃO

As transformações de visualização posicionam o observador e o sistema de coordenadas do olho no espaço do sistema de coordenadas do mundo.

Neste processo determina-se o sistema de coordenadas do olho.

As transformações de visualização dividem-se em: vista polar ("polarview") ou vista de cena ("lookat view").

Ambas vistas determinam um sistema de coordenadas do mundo de regra da mão direita com x à direita, y para cima e z em direção ao observador. As rotações também obedecem a regra da mão direita.

3.5.2.1 - VISTA POLAR

Define a posição do observador em coordenadas polares. Assume-se que o objeto que está sendo observado está posicionado na origem e que o observador (ponto de vista) está em qualquer posição do espaço do sistema de coordenadas do mundo.

Especifica-se a posição do observador com 4 parâmetros: a distância entre o observador e o objeto, o ângulo de incidência, o ângulo de Azimuth e o ângulo de torsão.

A distância pode ser vista na figura (Fig. 3.33) a seguir:

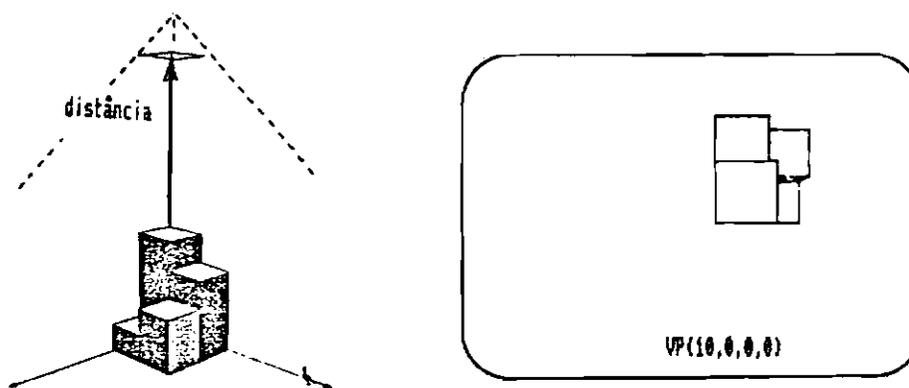


Fig. 3.33 - Distância entre o observador e a origem.
FONTE: Control Data Corporation (1986),
p. 50.

O ângulo de Azimuth fica no plano xy e é medido a partir do eixo y. Representa a diferença entre a orientação do objeto no eixo y (0 graus) e a orientação do observador no plano xy. A direção de rotação obedece a regra da mão esquerda.

Veja o ângulo de Azimuth na figura (Fig. 3.34) a seguir:

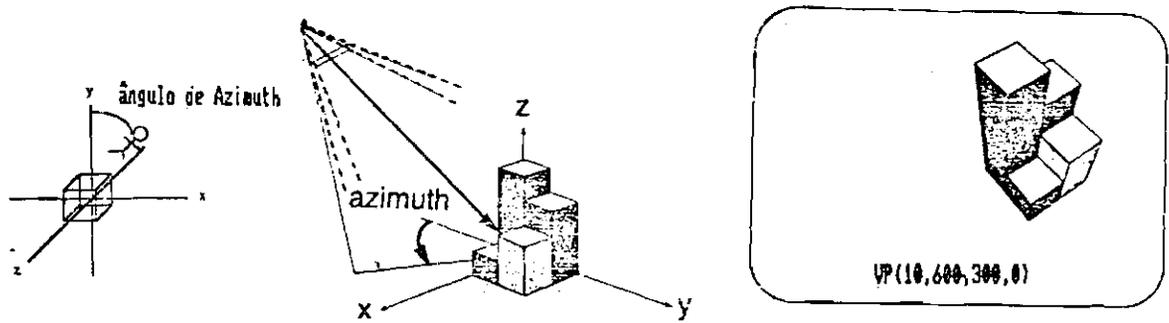


Fig. 3.34 - Ângulo de Azimuth.

FONTE: Control Data Corporation (1986), p. 51.

O ângulo de incidência fica no plano yz e é medido a partir do eixo z. Representa a diferença entre a orientação do objeto no eixo z (0 graus) e a orientação do observador no plano yz. A rotação também obedece à regra da mão esquerda. Observe a figura (Fig. 3.35) a seguir:

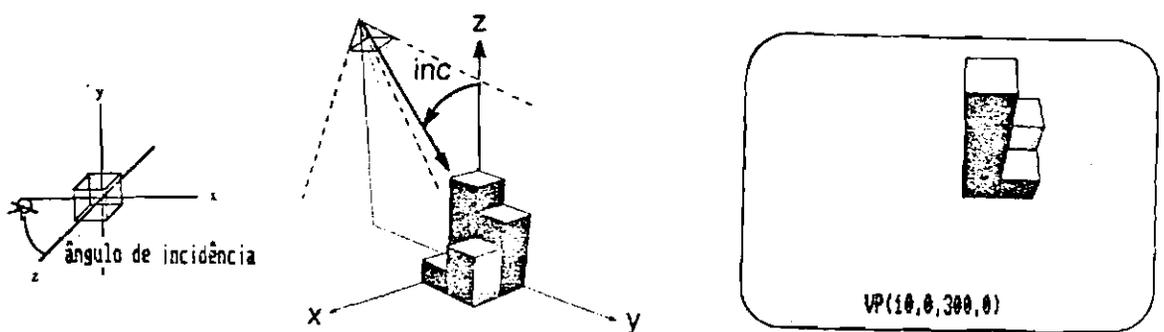


Fig. 3.35 - Ângulo de incidência.

FONTE: Control Data Corporation (1986), p. 52.

O ângulo de torsão é a quantidade de rotação, seguindo a regra da mão direita, em torno da linha de visualização.

A linha de visualização é uma linha reta entre o observador e o objeto.

Observe o ângulo de torsão na figura (Fig. 3.36) a seguir:

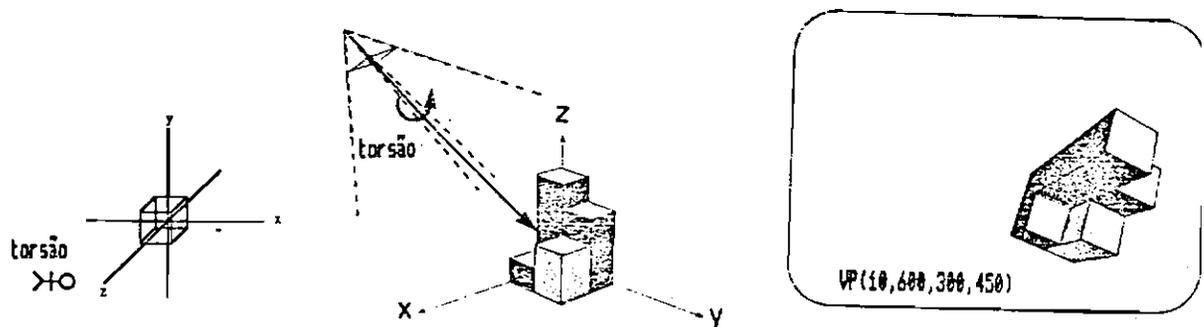


Fig. 3.36 - Ângulo de torsão.

FONTE: Control Data Corporation (1986), p. 53.

Representando a distancia por d , o ângulo de Azimuth por az , o ângulo de incidência por in e o ângulo de torsão por tw , determina-se a vista polar usando :

$$VP(d, az, in, tw) = R_z(-az)R_x(-in)R_z(tw)T(0, 0, -d) \quad (3.108)$$

R_z , R_x , e T , representam as matrizes de rotação e a matriz de translação apresentadas anteriormente.

3.5.2.2 - VISTA DE CENA

Estabelece um ponto de vista e um ponto de referência na linha de visualização no sistema de coordenadas do mundo.

Especifica-se a posição do observador (v_x, v_y, v_z), a localização do ponto que o observador está olhando (p_x, p_y, p_z) e um ângulo de torsão obedecendo a regra da mão direita.

Observe a figura (Fig. 3.37) a seguir:

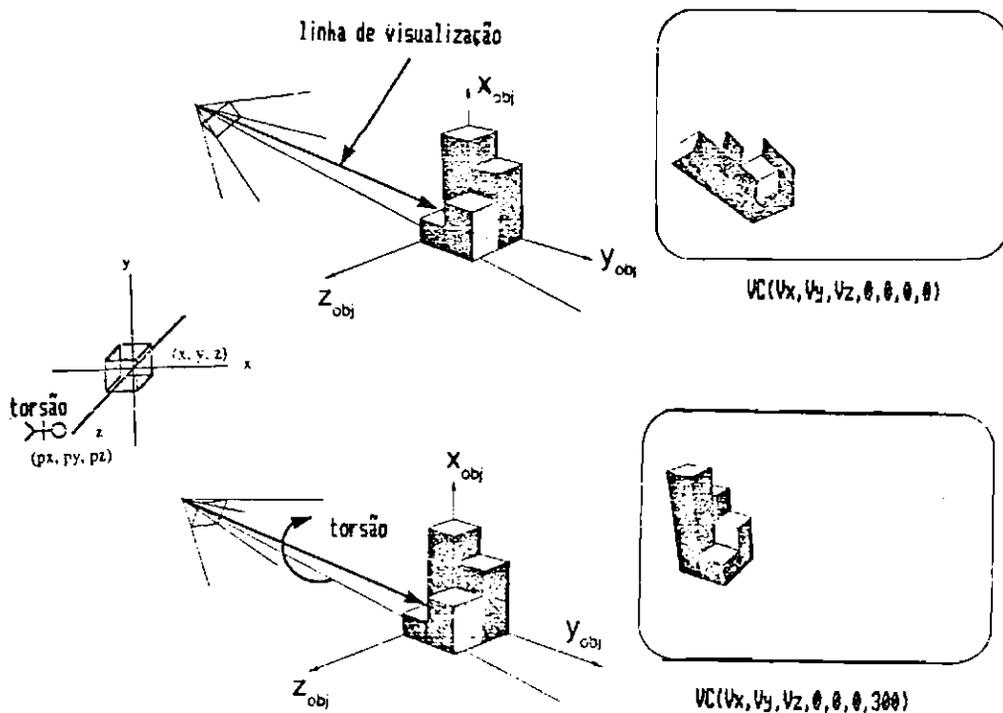


Fig. 3.37 - Visualização de cena.

FONTE: Control Data Corporation (1986), p. 54.

A visualização de cena pode ser calculada, usando como parâmetros, o ponto de vista (v_x, v_y, v_z) , o ponto de referência (p_x, p_y, p_z) e o ângulo de torsão tw , por:

$$VC(v_x, v_y, v_z, p_x, p_y, p_z, tw) = T(-v_x, -v_y, v_z) R_y(tw) R_x(tw) R_z(-tw) \quad (3.109)$$

onde:

$$\begin{aligned} \text{sen}(tw) &= (p_x - v_x) / ((p_x - v_x)^2 + (p_z - v_z)^2)^{0.5} \\ \text{cos}(tw) &= (v_z - p_z) / ((p_x - v_x)^2 + (p_z - v_z)^2)^{0.5} \\ \text{sen}(tw) &= (v_y - p_y) / ((p_x - v_x)^2 + (p_y - v_y)^2 + (p_z - v_z)^2)^{0.5} \\ \text{cos}(tw) &= ((p_x - v_x)^2 + (p_z - v_z)^2)^{0.5} / ((p_x - v_x)^2 + (p_y - v_y)^2 + (p_z - v_z)^2)^{0.5} \end{aligned} \quad (3.110)$$

3.5.3 - TRANSFORMAÇÕES DE PROJEÇÃO

Após descrever um modelo geométrico 3D, é necessário mostrá-lo na tela bidimensional. A representação do modelo na tela 2D chama-se projeção do modelo 3D.

Projetar objetos é análogo a tirar fotografias do objeto. Existe um objeto 3D que deseja-se representá-lo em uma superfície 2D, então o observador aproxima-se ou afasta-se para ajustar quanto do mundo do objeto (seu fundo) aparecerá na figura e uma vez decidido capta-se a cena em 2D.

Se o objeto está próximo do observador não aparecerá muito do fundo do objeto na cena, porém se o objeto está longe, o fundo do objeto ou seu mundo será maior.

Uma vez decidido quanto do fundo incluir, define-se um volume de visualização ("viewing volume") que pode ser de qualquer tamanho e onde estão incluídos o objeto e parte do seu mundo.

A forma do volume de visualização (que pode ser de qualquer tamanho) determina como que o objeto 3D é transformado em uma projeção 2D, a qual pode ser uma projeção em perspectiva ou uma projeção ortográfica.

As transformações de projeção definem o mapeamento do sistema de coordenadas do olho com a tela. O olho do observador é colocado em um sistema de regra de mão direita com o observador posicionado na origem olhando em direção do eixo $-z$.

Uma janela dentro da tela ("viewport") que especifica uma porção da tela onde o objeto será projetado, está associada com cada tipo de transformação de projeção.

3.5.3.1 - PROJEÇÃO EM PERSPECTIVA

Quando se observa através de uma janela um carro que está a 10 metros de distância e uma casa a 300 metros, dá a impressão que o carro pode passar por cima da casa sem muitos problemas. Isto é devido a que as pessoas olham em perspectiva. Um objeto que está perto sempre parece maior que outro objeto idêntico que está mais longe.

A projeção em perspectiva define um volume de visualização com forma de pirâmide truncada. Os objetos que se encontram mais perto ocupam maior volume de visualização do que outros objetos que estão mais afastados.

Observe a figura (Fig. 3.38) a seguir, onde mostra-se o volume de visualização em perspectiva:

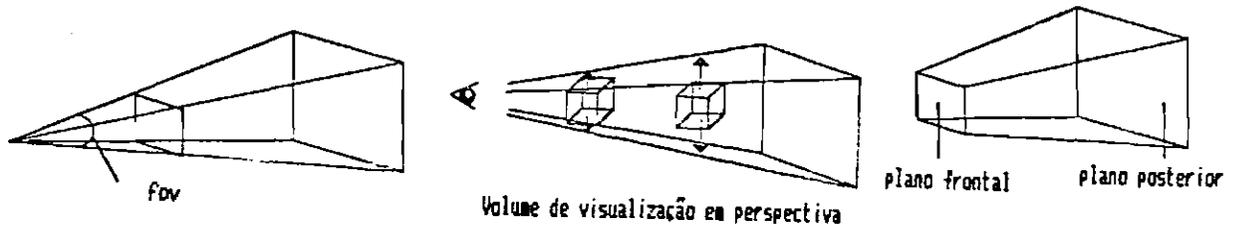


Fig. 3.38 - Volume de visualização em perspectiva.

FONTE: Control Data Corporation (1986),
p. 55.

Olhando a figura (Fig. 3.38) anterior, pode ser observado que a pirâmide é limitada por 6 planos ou planos de contorno ("clipping planes") e da mesma figura determinam-se os 4 parâmetros que determinam uma projeção em perspectiva:

1. O ângulo de visualização (fov), que é o ângulo que formaria o ápice da pirâmide se esta não fosse truncada.
2. A relação de aspecto (as), que é a relação de x para y . Afeta a localização dos planos de contorno superior, inferior, esquerda e direita.
3. As coordenadas dos planos frontal e posterior (nr e fr). Representam a distância da origem (ápice da pirâmide) a estes planos.

Com estes 4 parâmetros: fov, as, nr e fr, pode ser determinada a transformação em perspectiva de um ponto do modelo usando:

$$P(\text{fov}, \text{as}, \text{nr}, \text{fr}) = \quad (3.111)$$

$$\begin{vmatrix} \cot(\text{fov}/2)/\text{as} & 0 & 0 & 0 \\ 0 & \cot(\text{fov}/2) & 0 & 0 \\ 0 & 0 & (\text{nr}+\text{fr})/(\text{nr}-\text{fr}) & -1 \\ 0 & 0 & (2\text{nrfr})/(\text{nr}-\text{fr}) & 0 \end{vmatrix}$$

3.5.3.2 - PROJEÇÃO ORTOGRÁFICA (EM PARALELO)

A projeção ortográfica faz com que objetos que são do mesmo tamanho sejam vistos do mesmo tamanho, independente de estar próximos ou afastados.

Neste tipo de projeção, o volume de visualização é um paralelepípedo quadrado ou retângular. A quantidade de volume de visualização que qualquer objeto ocupa depende somente do seu tamanho atual.

O volume de visualização para este tipo de projeção pode ser visto na figura (Fig. 3.39) a seguir:

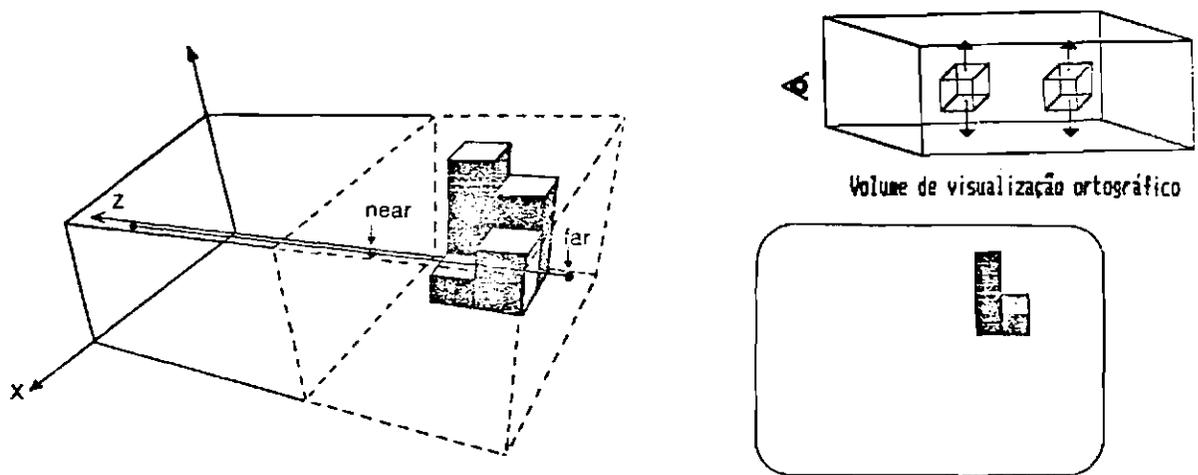


Fig. 3.39 - Volume de visualização ortográfico.

FONTE: Control Data Corporation (1986),
p. 56.

Na figura (Fig. 3.39) anterior, observa-se que este volume é limitado por 6 planos de contorno ("clipping planes"): topo, base, direita e esquerda (que determinam os planos xy) e frontal (o mais próximo da origem) e posterior que determinam os planos z .

Assim mesmo, *near* e *far* representam distâncias ao longo da linha de visualização e podem ser negativos. Portanto o olho do observador não necessariamente está na origem.

Chamando os planos topo, base, esquerda, direita, frontal e posterior de *to*, *bo*, *le*, *ri*, *nr* e *fr*, respectivamente, pode-se calcular a transformação ortográfica de um ponto usando:

$$O(l_e, r_i, b_o, t_o, n_r, f_r) = \quad (3.112)$$

$$\begin{vmatrix} 2/(r_i - l_e) & 0 & 0 & 0 \\ 0 & 2/(t_p - b_o) & 0 & 0 \\ 0 & 0 & 2/(n_r - f_r) & 0 \\ (l_e + r_i)/(l_e - r_i) & (b_o + t_o)/(b_o - t_o) & (n_r + f_r)/(n_r - f_r) & 1 \end{vmatrix}$$

3.6 - FUNDAMENTOS DE MODELAMENTO DE SÓLIDOS

Um modelo geométrico de um sólido é a representação não ambígua e a informação total da forma de um objeto físico em uma forma tal que o computador possa processar.

Modelamento de sólidos é um aspecto importante do modelamento geométrico que é usado para criar e comunicar informação a respeito da forma de um objeto sólido. Isto implica na construção e manutenção de um modelo do sólido para futuro acesso e análise.

Isto quer dizer que se A é um modelo de B, então A pode ser usado para responder questões acerca de B tais como propriedades volumétricas ou acerca de propriedades topológicas, tais como conectividade e relações de continência.

O sistema de modelamento de sólidos geralmente mantém dois tipos principais de dados para descrever o modelo, os quais são:

1. Dados geométricos

Que são os parâmetros básicos de definição da forma tais como os coeficientes geométricos de uma superfície bicúbica

2. Dados topológicos

Os quais incluem relações de conectividade entre os componentes geométricos.

O princípio dos sistemas para modelamento geométrico de objetos sólidos rígidos, reside na definição de estruturas de símbolos (representações) designando "sólidos abstratos" (subconjuntos do espaço euclidiano) que modelam sólidos físicos.

Estes sistemas geométricos têm 4 componentes primários como ilustra a figura (Fig. 3.40) abaixo:

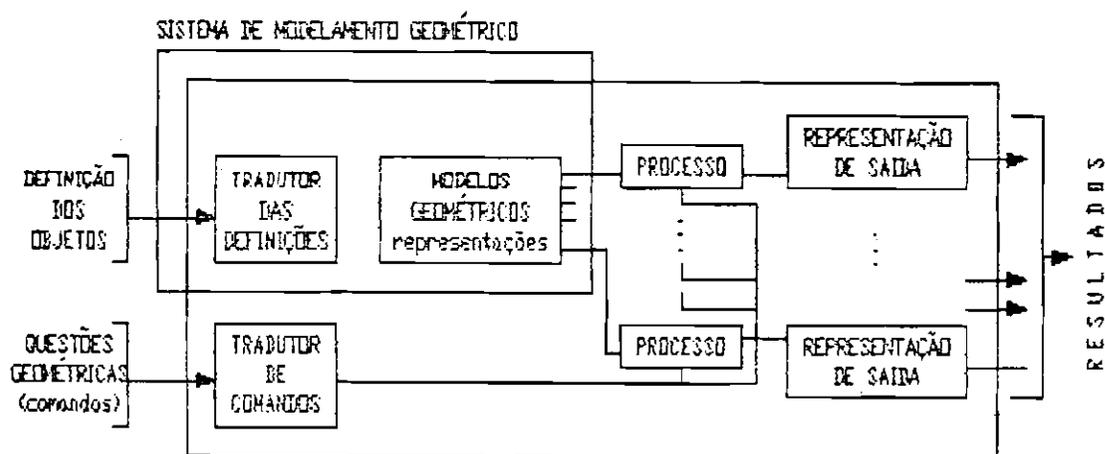


Fig. 3.40 - Sistema geométrico.

1. Estruturas simbólicas que representam objetos sólidos.
2. Processos que usam tais representações para responder questões geométricas a respeito dos objetos .
3. Facilidades de entrada, meios para criar e editar representações de objetos e para alterar processos existentes,
4. Facilidades de saída e representações de resultados.

O subsistema que proporciona facilidades para entrar, armazenar e modificar representações de objetos é chamado de Sistema de Modelamento Geométrico SMG.

A maioria dos SMG usados atualmente em CAD/CAM sofrem de deficiência representacional.

Especificamente, representações são coleções de curvas e superfícies as quais não precisam corresponder a objetos sólidos únicos e bem definidos.

Algoritmos geométricos não manipulam sólidos físicos, ao contrário manipulam dados (estruturas simbólicas) os quais representam sólidos.

Por exemplo, suponha que deseja-se representar um poliedro sólido de faces planas. Um recurso é se basear nos lados porque estes são as características mais perceptíveis do sólido e definem-se bem como retas. Cada lado pode ser definido por seus pontos extremos e o sólido pode ser representado como uma coleção de 6-tuplas de números reais:

$$(x_i, y_i, z_i, x_j, y_j, z_j) \quad (3.113)$$

Além de definir a sintaxe (notação) da representação, deve-se definir também a sua semântica (significado geométrico), especificando uma regra que associe geometria com representações.

Neste exemplo, interpreta-se (x_i, y_i, z_i) e (x_j, y_j, z_j) como as coordenadas de dois pontos e associa-se a 6-tupla com o segmento de reta cujos pontos extremos têm estas coordenadas.

Existem questões fundamentais as quais qualquer teoria de modelamento geométrico deveria atender:

1. Os contornos fornecem informação suficiente para uma computação automática das propriedades geométricas do sólido?
2. De que maneira, um algoritmo geométrico ou sistema pode determinar se está operando sobre dados válidos?
3. Existe mais de uma maneira de representar um sólido?
4. É possível determinar se diferentes representações correspondem ao mesmo sólido?
5. Podem certas propriedades geométricas ser computadas com uma representação e não com outra?
6. Qual representação é a melhor?

Inicialmente deve-se definir o que se entende por matemática de um sólido.

No modelamento geométrico postulam-se entidades geométricas abstratas, subconjuntos do espaço euclidiano E , para a modelagem de sólidos físicos.

A noção de sólido abstrato obedece as seguintes propriedades matemáticas:

1. Rigidez

Um sólido abstrato deve ter uma configuração ou forma invariante, a qual é independente da localização e orientação do sólido.

2. Tridimensionalidade homogênea

Um sólido deve ter um interior, e um contorno do sólido não pode ter porções isoladas ("dangling edges").

3. Finitude

Um sólido deve ocupar uma porção finita do espaço.

4. Fechamento sob movimentos rígidos e certas operações booleanas

Movimentos de corpo rígido (translação e/ou rotação) ou operações que adicionam ou removem material (operações booleanas regularizadas) quando aplicados a sólidos devem produzir outros sólidos.

5. Descrição finita

Deve existir um aspecto finito do sólido (por exemplo, um número finito de faces) para que este possa ser representado por comandos ou dados finitos no computador.

6. Determinação de contornos

O contorno (fronteira) de um sólido deve determinar sem ambiguidade o que está dentro do sólido e portanto confinar o sólido.

Os modelos adequados para sólidos são subconjuntos de E que são limitados (ou finitos, que ocupam uma porção finita no espaço), fechados (fechamento sob movimentos rígidos e operações booleanas), regulares e semi-analíticos; estes conjuntos são chamados r -sets.

Estes conjuntos são classes congruentes ou seja, são uma coleção de conjuntos que podem ser obtidos a partir de outros por sequências de translações e rotações.

As noções matemáticas de conjunto "regular fechado" e conjunto "semi-analítico", podem ser explicadas olhando os contraexemplos mostrados na figura (Fig. 3.41) a seguir:

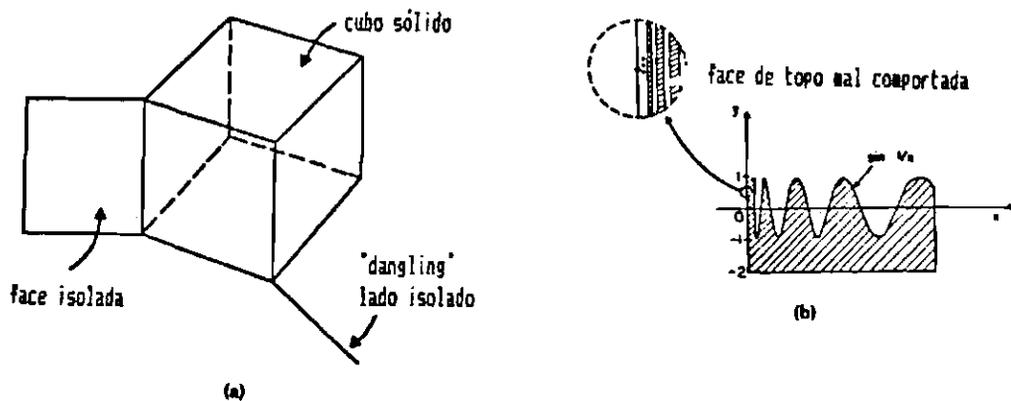


Fig. 3.41 - (a) Conjunto não regular; (b) conjunto não semi-analítico.

FONTE: Requicha (1977), p. 4.

O conjunto da figura (Fig. 3.41 (a)) acima é fechado, porém não é um conjunto regular fechado porque não é tridimensionalmente homogêneo, pois sua fronteira tem porções isoladas ("dangling") que não são adjacentes aos interiores do conjunto.

O conjunto da figura (Fig. 3.41 (b)) não é semi-analítico porque sua face de topo é mal comportada e oscila rapidamente conforme se aproxima da face esquerda.

Apresentam-se algumas propriedades formais de esquemas de representação de modelos (ER):

1. Domínio

Conjunto de objetos que o ER é capaz de representar. O domínio do esquema poderia incluir somente objetos com faces planas, convexos e poliedros simplesmente conectados.

O conjunto de formas válidas de representação definem a imagem do ER.

2. Completeza

É a medida da capacidade do modelo (e portanto do ER) em responder a questões geométricas. Por exemplo, modelos em arame ("wireframe") típicos são representações incompletas de um sólido pois não conseguem responder perguntas sobre normais de superfície ou volume de superfície.

3. Unicidade

Fator importante na determinação de igualdade de objetos. Por exemplo, representações repetidas podem ser afastadas do banco de dados somente se for possível determinar que duas representações no mesmo ER correspondem ao mesmo objeto.

Os ER que são tanto ambíguos como únicos, são altamente desejáveis porque são mapeamentos 1:1. Isto significa que representações distintas em tais esquemas correspondem a objetos distintos.

4. Consistência

Refere-se à quantidade de dados requeridos para definir um objeto em um ER particular. Quanto mais conciso é o modelo, mais conveniente é para armazenar ou transmitir, e menos dados redundantes este contém.

3.7 - ESQUEMAS PARA A REPRESENTAÇÃO DE SÓLIDOS RÍGIDOS

3.7.1 - INTRODUÇÃO

Apresentam-se os principais esquemas de representação que produzem representações não ambíguas de sólidos e que de uma maneira direta ou indireta foram relacionados com a representação dos sólidos manipulados pela estrutura de dados desenvolvida.

Existem 6 métodos importantes de construir modelos de sólidos:

1. Instâncias ou formas parametrizadas.
2. Decomposição celular
3. Representação por varredura ("sweeping")
4. Sólido geométrico construtivo CSG ("Constructive Solid Geometric")
5. Representação por contornos ("boundary")
6. Representação em arame ("wireframe")

Entre estes, serão descritos:

1. A decomposição celular

Método que descreve como um sólido complexo pode ser construído a partir de sólidos mais simples.

2. A representação por varredura

Por ter sido usado para construir os sólidos do nível 1 da hierarquia da estrutura de dados implantada ou folhas da árvore binária. Sólidos definidos como sólidos primitivos na estrutura.

3. A representação CSG

Por ser o método que descreve o processo de construção de um sólido a partir de sólidos mais simples seguindo uma estrutura em árvore binária usando como nós os operadores booleanos e transformações de corpo rígido. A estrutura hierárquica implantada segue este modelo.

4. A representação por arame

No sistema CAD usado, os sólidos são representados usando este modelo.

Servindo como base fundamental para o entendimento da representação CGS, apresenta-se inicialmente os conceitos de modelos booleanos, que descrevem as operações booleanas entre sólidos .

3.7.2 - MODELOS BOOLEANOS

É a representação booleana de um objeto sólido pela combinação de dois ou mais objetos simples.

Os símbolos U , \cap e $-$ (união, intersecção e diferença), denotam os operadores booleanos regularizados.

O termo regularizado é consequência da restrição de que a combinação booleana de dois ou mais sólidos dá como resultado um sólido válido, isto é, que possua as propriedades matemáticas de um sólido abstrato descritas na seção 3.6.

Como exemplo veja a figura abaixo (Fig. 3.42) que representa a combinação booleana de 3 sólidos: A, B e C, dando como resultado o sólido $D = (A \cup B) - C$:

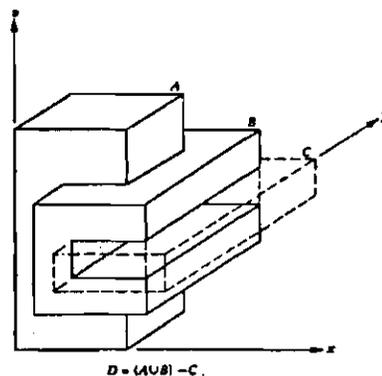


Fig. 3.42 - Modelo procedural simples.

FONTE: Mortenson (1985), p. 439.

Esta expressão booleana definindo D não diz nada a respeito da sua geometria ou topologia, apenas diz como construí-lo. Portanto o modelo booleano é dito representação procedural ou modelo não avaliado.

A árvore binária do sólido D é mostrada na figura (Fig. 3.43) a seguir:

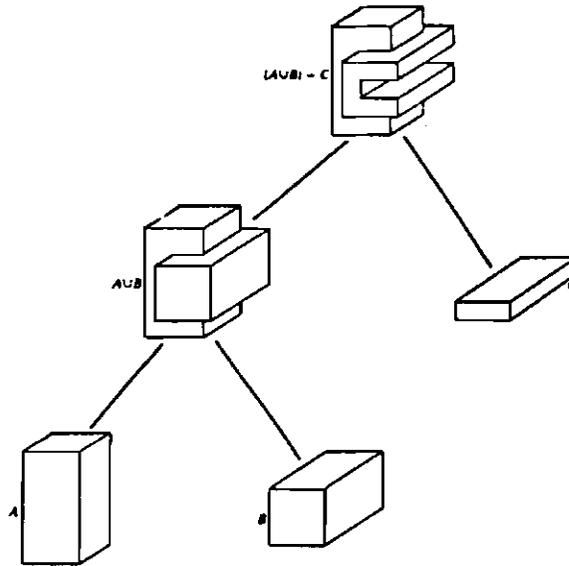


Fig. 3.43 - Árvore binária para $D=(A \cup B)-C$.
 FONTE: Mortenson (1985), p. 440.

Fazendo uma analogia com o trabalho, na figura (Fig. 3.43) acima, as folhas da árvore A, B e C representam os sólidos primitivos gerados por varredura.

O nó $(A \cup B)$ representa um sólido combinado construído a partir dos sólidos A e B, os quais na verdade são dois sólidos transformados pois foram transladados para ser posicionados e usar a união como operador booleano.

A raiz da árvore é o sólido final que além de representar outro sólido combinado, representa a última instância de sólido composto o qual é uma agregação de todos os sólidos que compõem o sólido final.

A figura (Fig. 3.44) a seguir, é outro exemplo de um modelo booleano de um objeto sólido.

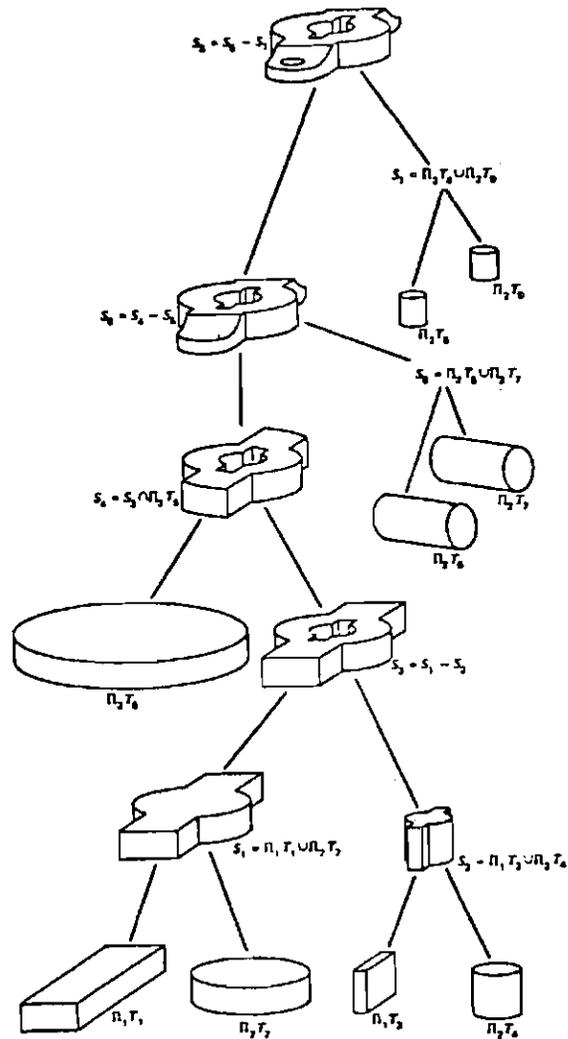


Fig. 3.44 - Modelo booleano de uma peça mecânica.
 FONTE: Mortenson (1985), p. 441.

Aqui Π_i denota um sólido, por exemplo, Π_1 denota um paralelepípedo retangular e Π_2 um cilindro. T_i representa uma transformação de corpo rígido (translação, rotação ou mudança de escala).

3.7.3 - DECOMPOSIÇÃO CELULAR

Este tipo de representação é citado, pois exemplifica como um sólido pode ter sido composto a partir de sólidos mais simples.

Considere um objeto sólido comum, por exemplo uma caneca como mostra a figura (Fig. 3.45) abaixo, e decomponha esta caneca em peças separadas, tal que cada peça da decomposição final é mais fácil de representar do que a caneca original.

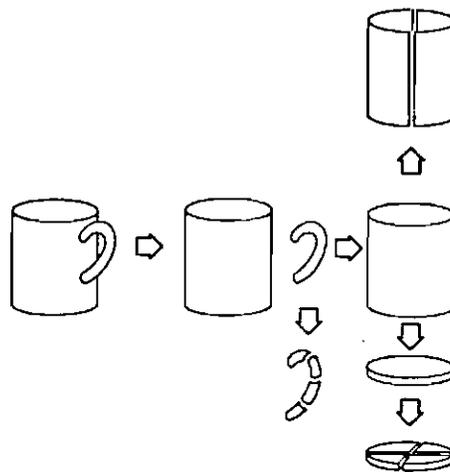


Fig. 3.45 - Decomposição celular.

FONTE: Mortenson (1985), p. 451.

Este processo é chamado decomposição celular, pois qualquer sólido pode ser representado como a soma ou união de um conjunto de células que dividem o sólido.

No trabalho desenvolvido a composição segue a ordem inversa a este tipo de representação, porém, a busca, atualização, inserção e remoção de um determinado sólido componente do sólido final na estrutura do banco de dados segue esta filosofia de decomposição.

3.7.4 - REPRESENTAÇÃO POR VARREDURA (REPRESENTAÇÃO "SWEEP")

Este tipo de representação foi a escolhida para a construção dos sólidos primitivos ou folhas da árvore da estrutura hierárquica implantada no sistema, em virtude da sua facilidade de aplicação, flexibilidade de uso e a disponibilidade desta ferramenta na biblioteca gráfica do sistema.

A representação por varredura é baseada no conceito de mover um ponto, curva, superfície ou sólido em torno de uma trajetória que pode ser um eixo de rotação (varredura rotacional) ou ao longo de um vetor de translação (varredura translacional) definindo um objeto uni, bi ou tridimensional.

O termo gerador é usado para denotar o objeto e o termo diretor para denotar a trajetória.

A figura (Fig. 3.46) a seguir, ilustra exemplos de varredura translacional e rotacional:

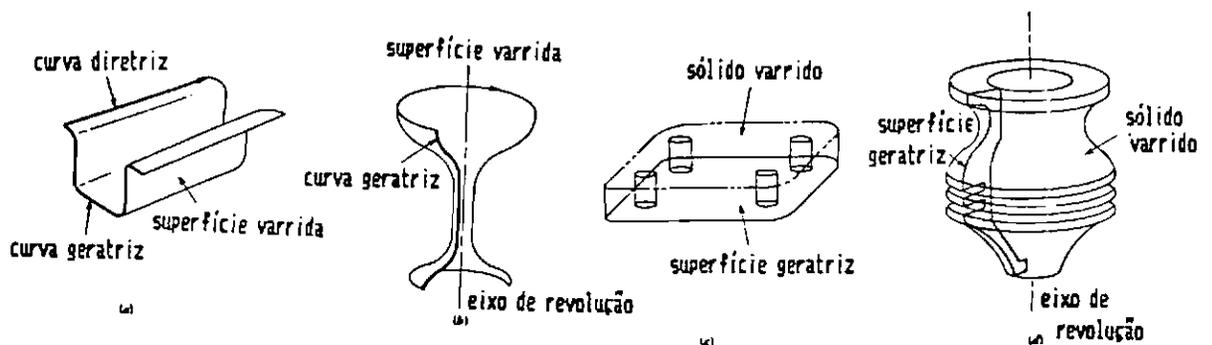


Fig. 3.46 - Exemplos de representação por varredura.

FONTE: Mortenson (1985), p. 456.

A curva diretriz não é necessariamente um elemento do objeto varrido, e no caso da varredura rotacional implica em um algoritmo para mover cada ponto da geratriz ao longo de um arco circular contido em um plano perpendicular ao eixo de rotação e com um raio definido como a distância perpendicular do ponto ao eixo.

Em todos os casos, a forma do objeto (perfil) que está sendo varrido não muda.

Existem maneiras óbvias e não tão óbvias de criar objetos dimensionalmente não homogêneos (propriedade matemática que define que um sólido deve ter um interior finito bem definido e que o contorno do sólido ou de um objeto 2D não deve ter porções isoladas).

Observe a figura (Fig. 3.47) a seguir:

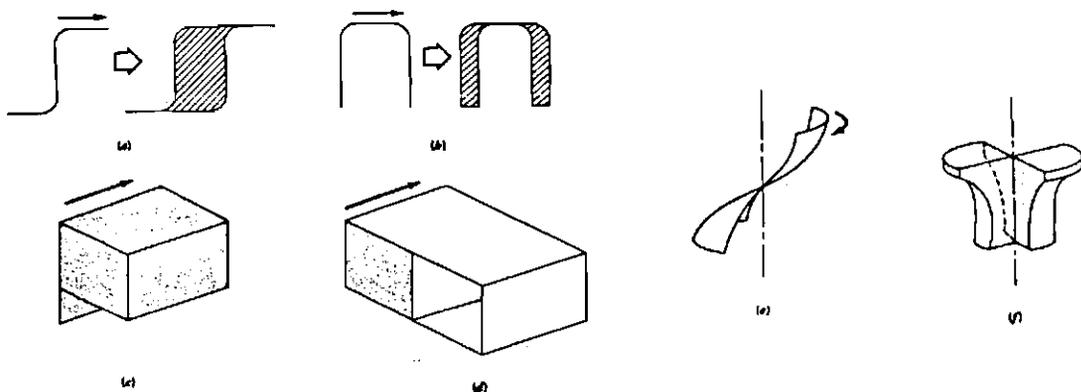


Fig. 3.47 - Representações por varredura dimensionalmente não homogêneas.

FONTE: Mørtenson, (1985) p. 457.

Em (a) (figura (Fig. 3.47) acima), a varredura translacional da curva que cria a superfície, também cria 2 arestas isoladas ("dangling"). Em (b) são criadas duas regiões bidimensionais unidas por uma estrutura unidimensional. Em (c) e (d) as geratrizes inválidas ou não homogêneas criam sólidos dimensionalmente não homogêneos e ambiguidades. E em (e) e (f) a varredura rotacional de uma geratriz que é atravessada pelo eixo de rotação, produz uma superfície ou sólido com uma singularidade.

Estas condições produzem resultados inaceitáveis; porém para sistemas de modelamento que precisam altos graus de liberdade, como arte por computador, estes resultados podem ser desejáveis.

Loosing e Eshleman (Mortenson, 1985), desenvolveram uma técnica para representar objetos de secção transversal constante usando uma trajetória geratriz curva e orientada de 6. componentes chamada "curva PD" (curva de posição e direção), através da qual a secção transversal é varrida.

Esta curva PD geralmente é uma curva cúbica paramétrica que continuamente especifica sua posição e orientação no espaço.

Os três primeiros componentes da curva definem uma equação cúbica paramétrica e contínua de posição no espaço 3D e as outras três definem uma equação cúbica paramétrica correspondente a um vetor com uma direção associada. Uma variável paramétrica comum associa o vetor direcional com uma posição específica sobre a curva.

A curva PD define um sistema de coordenadas curvo e retorcido ("twisted"), o qual, olhando a figura (Fig. 3.48) abaixo, é computado da seguinte maneira:

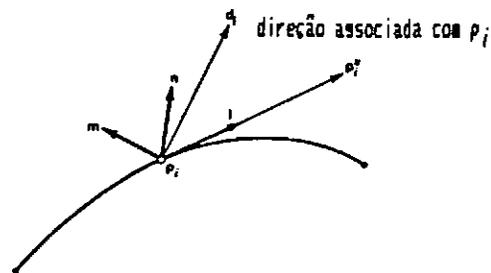


Fig. 3.48 - Características de uma curva PD.
FONTE: Mortenson (1985), p. 459.

Localiza-se a origem em um ponto p_i , é calculado p_i^u o vetor tangente, de p_i^u e d_i (d_i é dado por outra equação pc) acham-se os vetores ortogonais unitários l , m e n , tal que:

$$l = p_i^u / |p_i^u|$$

$$m = (d_i \times p_i^u) / |d_i \times p_i^u| \quad (3.113)$$

$$n = l \times m ,$$

os eixos l e n definem um plano de direção no qual d_i está localizado. Este plano de referência muda conforme o vetor tangente e d_i mudam continuamente ao longo da curva PD.

A curva PD define uma transformação contínua para pontos de uma curva de perfil ou de um objeto a ser varrido.

O esquema de representação por varredura é não ambíguo porém não único e seu domínio está limitado a objetos com simetria rotacional ou translacional.

No caso específico do sistema, o perfil deve ser um que defina uma figura bidimensional fechada ou contorno fechado, o qual pode ser composto a partir de retas, arcos ou B-Splines.

Uma exceção, é o caso do círculo que também faz parte da implementação, e que já é um contorno fechado. Um caso análogo, é o da curva B-Spline onde o primeiro ponto é unido ao último ponto definindo um contorno fechado.

Outra consideração, é que o eixo de rotação nunca atravessar do contorno fechado, porém o eixo de translação pode atravessar o contorno pois este representa apenas uma direção.

A função da biblioteca gráfica que permite a utilização do recurso da varredura, checa automaticamente a validade do sólido resultante, ou seja, se o eixo está bem definido e se o contorno é um contorno fechado, não permitindo que seja construído um sólido inválido ou sólidos dimensionalmente não homogêneos, como explicado acima.

3.7.5 - SÓLIDO GEOMÉTRICO CONSTRUTIVO CSG ("CONSTRUCTIVE SOLID GEOMETRIC")

CSG é um método de modelamento que define sólidos complexos como composição de sólidos simples ou primitivas usando operadores booleanos para executar esta composição. Também é conhecido como geometria de construção de blocos.

Requicha (1980), define CSG como uma generalização da decomposição celular. Nos modelos de decomposição celular, as células individuais são combinadas usando uma forma limitada do operador de união ("glue") o qual apenas une componentes cujas faces casam perfeitamente.

Os operadores CSG são mais versáteis desde que os contornos dos componentes unidos não precisam casar e seus interiores não precisam ser disjuntos (os interiores não se tocam).

Além disso CSG usa todos os operadores booleanos regularizados: \cup , \cap e $-$, tal que pode-se adicionar ou remover material.

Representações CSG de objetos, são árvores binárias ordenadas cujas folhas, ou são sólidos simples, ou dados de transformação para movimentos de corpo rígido (translação e/ou rotação). Os nós não terminais são, ou os operadores booleanos regularizados, ou movimentos de corpo rígido que operam nos seus dois subnós (ou sólidos de nível hierárquico menor).

Cada subárvore de um nó (que não seja uma folha de transformação) representa um sólido resultante das operações de combinação e transformação.

Como exemplo, observe-se a figura (Fig. 3.49) abaixo:

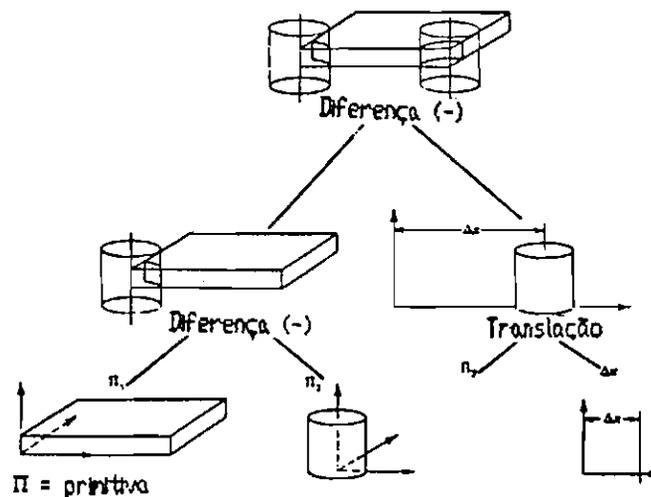


Fig. 3.49 - Representação CSG.

FONTE: Mortenson (1985), p. 462.

Na figura (Fig. 3.49) anterior, as 4 folhas representam as primitivas ou sólidos simples Π_1 e Π_2 , e a translação Δx . Os dois nós internos representam resultados das operações $(\Pi_1 - \Pi_2)$ e $\Pi_2 \cdot \Delta x$. A raiz representa o objeto final.

Deve ser observado que os objetos primitivos e intermediários são sólidos contornados válidos. Além disso, as transformações não estão limitadas a movimentos rígidos (translação e/ou rotação) podendo incluir transformações de escala e simetria.

Se as primitivas são sólidos contornados válidos e se os operadores de combinação são regularizados, então os modelos dos sólidos resultantes também são válidos e contornados. Não obstante, um sistema de modelamento pode permitir que o usuário defina suas próprias primitivas, porém o usuário ou o sistema deve verificar a validade do modelo.

Em geral, os sistemas de modelamento contemporâneos, oferecem um conjunto de primitivas concisas e compactas cujo tamanho, forma, posição e orientação são determinados por um conjunto de parâmetros especificados pelo usuário. As primitivas mais comuns são o cilindro, o cubo, a esfera, o cone e o toro de revolução.

Os sistemas de modelamento sofisticados, em geral geram duas representações do sólido:

1. Representação construtiva

Representada por uma estrutura de dados em árvore binária ligando primitivas e subsólidos subsequentes e usando operadores de transformação e combinação.

2. Representação dos contornos

A qual descreve as faces, arestas e vértices dos contornos do sólido. Esta descrição de contornos por si mesma se divide em duas formas:

- descrição da representação topológica da conectividade dos elementos de contorno,
- dados numéricos, descrevendo a geometria da forma e a posição destes elementos

A forma construtiva geralmente é chamada de um modelo procedural não avaliado e a forma de representação de contornos é chamada de um modelo avaliado.

O sistema Computervision usado na implementação também oferece um conjunto de primitivas básicas (objetos tridimensionais) definidas a partir de certos parâmetros assim como gera as duas representações do sólido citadas acima.

Porém, no modelo desenvolvido, o usuário não usa este conjunto de primitivas, ele deve construir suas próprias primitivas pela varredura de um contorno fechado.

As operações booleanas (união, intersecção e diferença) assim como as operações de transformação (translação, rotação e mudança de escala) usadas para implementar o modelo, também fazem parte da biblioteca gráfica do sistema. A transformação de simetria não é considerada.

3.7.6 - REPRESENTAÇÃO EM ARAME (REPRESENTAÇÃO "WIREFRAME")

Os modelos em arame representam as arestas de um objeto e consistem de pontos, linhas e curvas. Na modelagem de sólidos tridimensionais usando representação em arame, existem os seguintes problemas:

1. A possibilidade de criar modelos ambíguos
2. A possibilidade de criar objetos absurdos
3. Deficiência de coerência visual ou gráfica para determinar os contornos
4. Possibilidade de redundância na definição de dados

Vários modelos em arame são não ambíguos (representação única) na representação de sólidos, tais como os modelos de 2 1/2 dimensão.

Por exemplo, os modelos da figura (Fig. 3.50) a seguir, podem ser interpretados de apenas uma maneira:

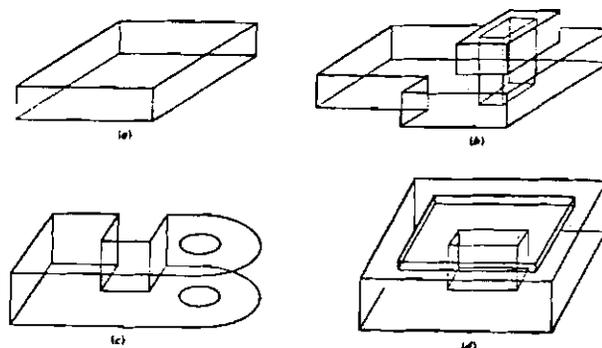


Fig. 3.50 - Modelos em arame de 2 1/2 dimensão não ambíguos.

FONTE: Mortenson (1985), p. 479.

Porém, a figura (Fig. 3.51) abaixo, é o exemplo clássico de um modelo em arame ambíguo, pois pode ser interpretado de varias maneiras:

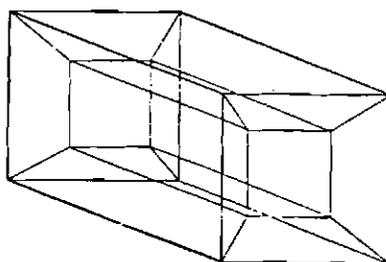


Fig. 3.51 - Ambigüidade do modelo em arame.

FONTE: Mortenson (1985), p. 480.

Se na figura (Fig. 3.50 (a)) uma das arestas apagada, então o resultado é um objeto topologicamente absurdo (sem sentido). Assim mesmo, na figura (Fig. 3.50 (c)), é difícil determinar os contornos.

Portanto o sistema em arame ainda é um modelo em evolução.

No sistema utilizado, os sólidos são representados em arame, porém com um algoritmo de remoção de linhas escondidas somado à disponibilidade de rotinas de cor e sombreamento ("shading"), é possível visualizar o sólido como um sólido do mundo real.

CAPÍTULO 4

DESCRIÇÃO DO SISTEMA CAD UTILIZADO

4.1 - INTRODUÇÃO

O presente trabalho foi desenvolvido em um sistema CAD/CAM Computervision's Designer V-X System.

Este sistema CAD possui um software próprio, o CADDs 4X, o qual representa um pacote gráfico, que oferece uma ampla variedade de funções gráficas para o modelamento de sólidos.

A linguagem VARPRO 2, também exclusiva do sistema, permite implementar programas de aplicação usando o pacote gráfico CADDs 4X. O modelador geométrico implementado neste trabalho foi desenvolvido na linguagem VARPRO 2.

CGOS 200X, um dos sistemas operacionais que rodam no sistema, foi o ambiente operacional da linguagem VARPRO 2 usada.

Apresenta-se uma descrição completa do sistema enfocando o hardware, o software e os dois ambientes de operação: o nível do sistema operacional OS e o nível do ambiente gráfico CADDs

4.2 - SISTEMA CAD/CAM COMPUTERVISION DESIGNER V-X

Entende-se por CAD/CAM como Computer-Aided Design/Computer-Aided Manufacturing (desenho auxiliado por computador/manufatura auxiliada por computador).

Um sistema CAD/CAM se baseia na capacidade do computador de guardar grandes quantidades de informações e processá-las rápida e precisamente.

Na figura (Fig. 4.1) abaixo, estão mostrados os componentes do sistema utilizado:

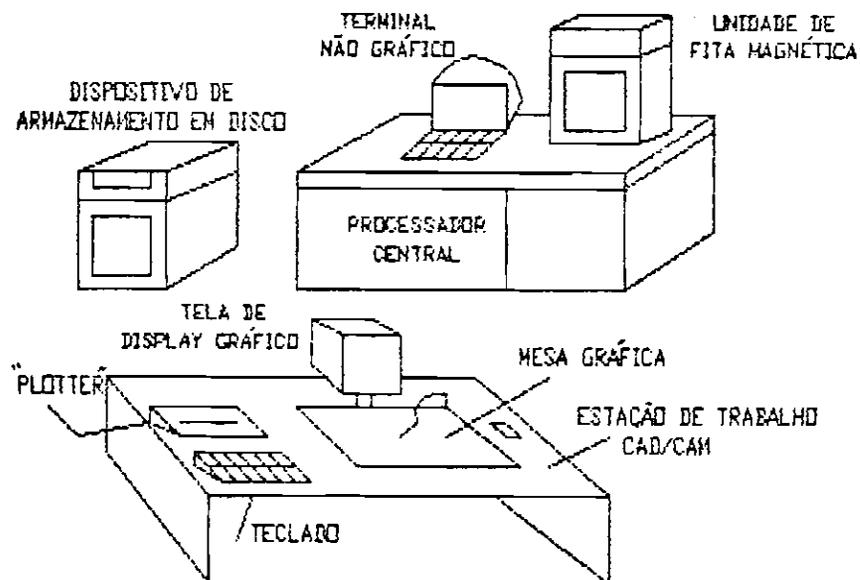


Fig. 4.1 - Componentes de um sistema CAD/CAM.

O sistema permite que o usuário entre com informações de desenho no sistema por meio de uma caneta de luz e mesa gráfica ou um teclado, para criar e/ou modificar interativamente desenhos do modelo o qual é mostrado na tela gráfica. O computador logo guarda o modelo em um banco de dados comum, o qual pode ser acessado por qualquer usuário.

4.2.1 - HARDWARE DO SISTEMA

O hardware do sistema CAD/CAM é composto por:

1. Processador

O processamento da informação é compartilhado com vários processadores que operam ao mesmo tempo e cada um com uma tarefa específica.

2. Dispositivo de armazenamento em disco

Para o armazenamento do software do sistema e informação relacionada a arquivos de trabalho ("part") e modelos geométricos desenhados no sistema.

3. Unidade de fita magnética

Permite armazenar e recuperar a informação do disco.

4. Estação de trabalho ("workstation")

O usuário cria desenhos em um arquivo de trabalho, usando a estação de trabalho, a qual é dividida nas seguintes partes:

A. Tela de display gráfico

De tecnologia "raster", de 24 linhas por 80 colunas, 1024 x 1024 "pixels" (elementos de imagem), modo gráfico e modo de texto ou modo combinado usando as 4 últimas linhas da tela em modo texto para interface com o sistema.

Possibilita texto de variáveis de estado ("status"), no canto superior esquerdo da tela o qual inclui:

- Nome do arquivo de trabalho ativo
- Nome da folha de desenho ("draw") em uso
- Modo operacional gráfico em uso (modo Model ou modo Draw)
- Plano de construção ativo em uma das vistas do arquivo de trabalho
- Número do "layer" (folha transparente)
- Unidade de medida usado pelo modelo

Podem ser usadas até 64 cores a partir das 7 cores básicas; para sombreamento podem ser definidas cores arbitrárias fornecendo porcentagens de quantidades das 3 cores fundamentais (vermelho, verde e azul).

Podem ser definidos 24 pontos de iluminação (focos) para sombreamento, assim como a cada "layer" associar uma das 7 cores básicas.

B. Mesa gráfica

Inclui caneta eletrônica e menus de mesa

C. Teclado alfanumérico

D. Plotter e impressora.

4.2.2 - SOFTWARE DO SISTEMA

O software do sistema está dividido em 2 níveis:

1. OS ("Operating System"): software do sistema operacional
2. CADDs: ambiente gráfico usando VARPRO2, uma linguagem gráfica de alto nível.

O software do sistema operacional, nível OS, é um nível não gráfico, que permite manipular arquivos de dados, editar texto e executar programas do usuário.

CADDs 4X (Computervision Automated Design and Drafting System) é o software que processa todas as operações gráficas suportadas pelo sistema, e está dividido em dois modos distintos de atividade:

1. MODEL: construção do modelo

2. DRAW: detalhe do modelo

A figura (Fig. 4.2) abaixo, ilustra o software do sistema:

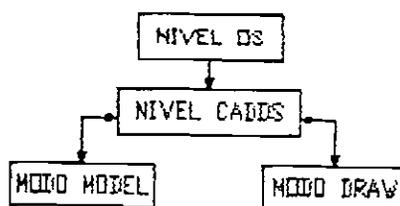


Fig. 4.2 - Software do sistema.

4.2.3 - NÍVEL OS

É um sistema operacional orientado a gráficos, multiusuário, multiprogramável, incluindo operações de entrada e saída assim como um sistema de gerenciamento de arquivos.

Suporta todas as aplicações gráficas interativas do nível CADD3, vários editores de texto próprios assim como várias linguagens de programação padrão como Fortran e Pascal as quais não tem acesso ao nível CADD3, pois existe a linguagem VARPRO2 a qual é dedicada a gráficos e exclusiva deste nível.

Permite até 10 usuários por "task" (tarefa), onde cada usuário possui um código de senha para entrar no sistema ("login" e "password").

4.2.4 - NÍVEL CADD5

4.2.4.1 - MODOS MODEL E DRAW

O CADD5-4X é um pacote gráfico que pode ser programado usando a linguagem gráfica VARPRO2 a qual tem sintaxe própria. Está dividido em dois modos: Model e Draw.

O modo Model é usado para criar uma representação do mundo real de um objeto em tres dimensões, criando a descrição e geometria do modelo. Opcionalmente também pode criar geometria de desenho em duas dimensões.

O modo Draw usa o modelo criado no modo Model, que está no banco de dados, para criar desenhos detalhados bidimensionais no modelo, tais como inserir dimensões, texto, etc.

Observe a figura (Fig. 4.3) a seguir, onde são diferenciados estes conceitos:

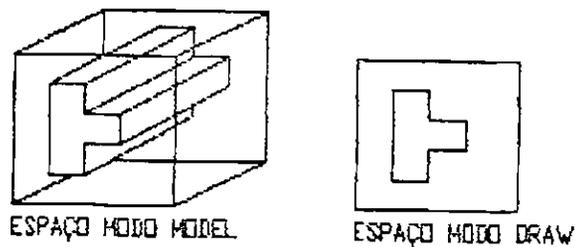


Fig. 4.3 - Espaços dos modos Model e Draw.

Na figura acima observa-se que o espaço do modelo Model é definido por um sistema de coordenadas tridimensional enquanto que o do modo Draw é um sistema bidimensional.

CADDS 4X permite criar vários desenhos ("draws" ou folhas de desenho) de múltiplas vistas 2D de um modelo 3D. Uma vista representa uma posição de referência onde a visão do observador está colocada, tal que o modelo pode ser visto de cima, de lado, de frente, etc.

A escala da folha de desenho usada, pode ser uma escala padrão disponível (tamanho A, B, C, D, E, AO, A1, A2, A3 ou A4) ou especificada pelo usuário usando uma determinada unidade de medida.

Uma folha de desenho é uma coleção de vistas, as quais descrevem o arquivo de trabalho (este armazena os modelos).

Podem ser criadas bastantes folhas de desenho (onde cada uma destas pode ter qualquer número de vistas com qualquer tamanho selecionado e um nome associado) no modo Draw e serem associadas a um modelo específico do modo Model.

Quando o modo Draw está ativo para editar a aparência visual do modelo, as mudanças que são feitas somente afetam a vista particular na qual a modificação gráfica é realizada. Isto é um contraste com o modo Model onde mudanças em uma vista são refletidas em todas as vistas pois o modelo é afetado.

Na verdade, inicialmente define-se uma ou várias vistas do modelo; posiciona-se esta vista ou cada uma destas em uma determinada posição da tela do vídeo (pois é possível olhar simultaneamente várias vistas de um modelo na tela), para logo selecionar um "draw" que é como colocar uma folha de desenho que cobre toda a tela, e escrever por cima do modelo como se fosse uma máscara sobreposta que não afeta o modelo, apenas está se escrevendo na folha de desenho.

A figura abaixo (Fig. 4.4) ilustra estes conceitos, onde se tem um modelo tridimensional e várias folhas de desenho associadas, cada folha contento qualquer número de vistas com qualquer escala selecionada:

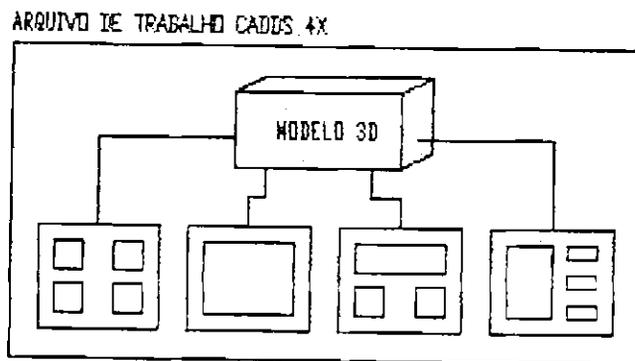


Fig. 4.4 - Modelos e folhas de desenho.

O espaço do modo Model (espaço Model) é um sistema de coordenadas 3D-xyz onde a origem dos 3 eixos é definido pelo usuário.

O espaço Draw é um sistema de coordenadas 2D xy onde a origem dos eixos é localizado no canto inferior esquerdo do vídeo.

Estes espaços estão mostrados na figura (Fig. 4.5) abaixo:

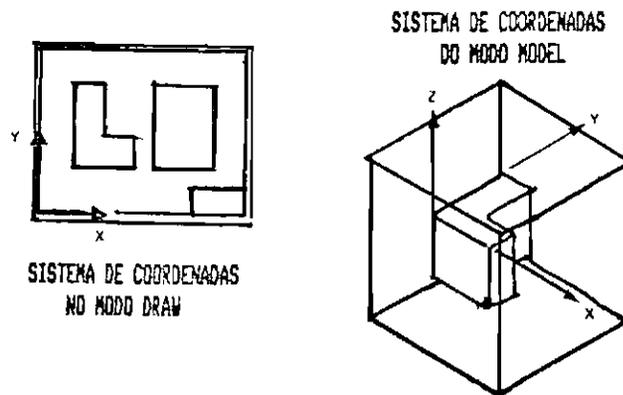


Fig. 4.5 - Sistema de coordenadas Model e Draw.

FONTE: Computervision Corporation (1984),
p. 8.

Toda sessão do usuário, isto é, toda parte que o usuário ativa, deve ser inicializada definindo-se a unidade de medida a ser considerada (m, cm, mm, pol, etc.) e ativando uma folha de desenho, pois se esta não é ativada, não é possível definir nenhuma vista e portanto o modelo não pode ser visto.

4.2.4.2 - PLANOS DE CONSTRUÇÃO

O conceito de plano de construção PC, é uma característica que proporciona uma visão real em CAD para o desenhista.

Um plano de construção é um plano que pode ser selecionado da geometria do modelo e ser pré-definido porque está associado à uma vista específica. Por exemplo: para a vista frontal, o plano de construção é frontal.

Uma vez definido e ativado um plano de construção, podem ser executados os seguintes tipos de atividade:

- todas as digitações na mesa gráfica, usando a caneta de luz, são projetadas diretamente no plano de construção ativo;
- um sistema de coordenadas é automaticamente estabelecido com a mesma origem que o plano de construção definido pelo usuário;
- existe um recurso que permite visualizar ou não os eixos do sistema de coordenadas;
- a unidade de medida deste sistema de coordenadas é a mesma que a especificada ao iniciar a parte.

Só se pode ativar um plano de construção por vez. Além dos planos de construção pré-definidos associados às vistas, é possível definir outros planos tomando como referência estes planos. Por exemplo, um plano paralelo ao plano frontal ou 45 graus em relação ao plano de topo.

Dadas as unidades de medida dos eixos do sistema de coordenadas, pode-se ligar no plano de construção uma grade de pontos que ajuda a visualizar as unidades de medida.

A distância entre os pontos desta grade, é um parâmetro que assume as unidades de medida adotados. Uma vez que a grade está lidada, todas as digitações podem ou não interpolar a estes pontos.

Esta grade de pontos é conhecida como "grid", e é colocada sobre o plano de construção ativo.

Assim, todo parâmetro que o usuário entra via teclado ou via caneta de luz, assume como padrão as unidades de medida definidas. Por exemplo, se as unidades em uso são milímetros, então um círculo de raio 10 será um círculo de 10 mm de raio no modelo.

4.2.4.3 -VISTAS

O conceito de vista é o de uma janela através do qual o modelo ou parte de um modelo pode ser visto.

CADDS 4X oferece um meio de definir a orientação do modelo (e portanto a vista) através do uso dos planos de construção.

Para definir a orientação de um modelo dentro de uma vista, pode ser usado um dos 7 planos de construção pré-definidos, cada um dos quais tem 7 orientações (vistas) associadas, das quais 6 são vistas ortográficas e a sétima é a vista isométrica:

PC TOPO	No. 1	VISTA TOPO (POR CIMA)
PC FRONTAL	No. 2	VISTA FRONTAL
PC DIREITA	No. 3	VISTA LATERAL DIREITA
PC BASE	No. 4	VISTA POR BAIXO
PC ESQUERDA	No. 5	VISTA LATERAL ESQUERDA
PC POSTERIOR	No. 6	VISTA POSTERIOR (POR TRÁS)
PC ISOMÉTRICO	No. 7	VISTA ISOMÉTRICA

A figura (Fig. 4.6) abaixo, mostra a relação entre o modo Model e o modo Draw em relação às vistas:

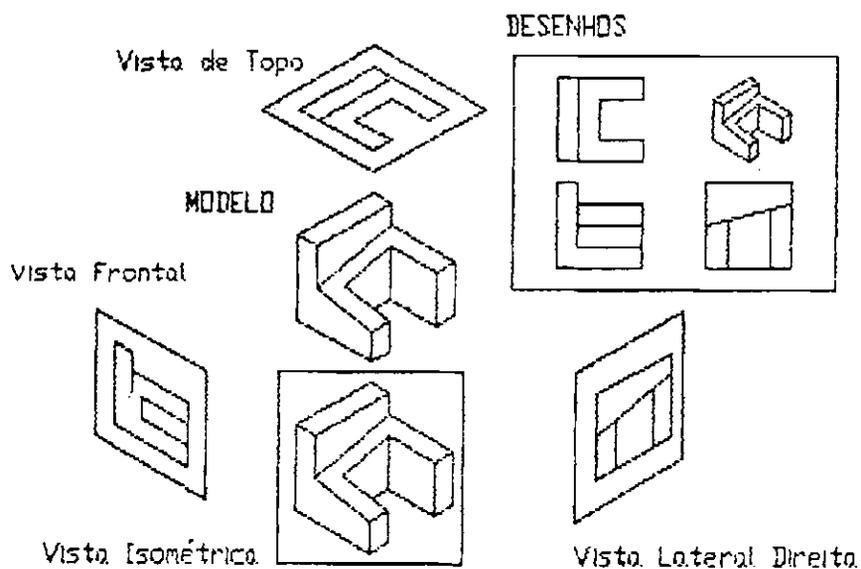


Fig. 4.6 - Modelos, vistas e folhas de desenho.

Aqui o modelo mostrado é tridimensional, está mostrado isometricamente e todas as vistas enquadram o modelo em orientações específicas.

Outro recurso que ajuda a visualizar os limites de cada vista dentro de uma folha de desenho, é ligar um retângulo que limita cada vista. Este retângulo é conhecido como "frame".

O modelo e suas folhas de desenho correspondentes, formam uma "part file", ou arquivo de trabalho, que é um arquivo com entidades gráficas construído no nível CADDs.

4.2.4.4 - "LAYERS"

O conceito de "layers" pode ser imaginado como sendo folhas transparentes ou níveis de trabalho, nas quais o usuário atribui tipos específicos de entidades.

Cada arquivo de trabalho tem 255 níveis de trabalho associados. Cada um dos quais, pode ser acessado diretamente e ter um tipo específico de entidade contido nele, assim como qualquer número destes podem ser mostrados ao mesmo tempo.

Pode-se associar a cada nível uma cor; assim as entidades contidas nele terão a cor associada. Isto é útil quando tendo vários níveis mostrados ao mesmo tempo, pela cor do nível possam ser diferenciadas quais entidades pertencem a quais níveis.

Existe um conceito sutil na relação de folhas de desenho, vistas e níveis de trabalho.

Todos as folhas compartilham estes 255 níveis e enfocam o mesmo modelo, o que diferencia uma folha da outra, são as vistas de cada folha. Todas as vistas da folha são as mesmas para todos os níveis da folha.

Como exemplo, suponha que temos 2 folhas: a folha A com 2 vistas e a folha B com 3 vistas, e exista uma entidade no nível 10. Estando a folha A ativa, no nível 10 será vista a entidade em 2 vistas, porém ativando a folha B e selecionando o nível 10, estará a mesma entidade mais agora vista através de 3 vistas.

4.2.4.5 - ENTIDADES GRÁFICAS

CADDS-4x permite ao usuário usar entidades gráficas pré-definidas para construir a geometria de um modelo entre as quais temos: pontos, retas, arcos, círculos, Splines, B-Splines, elipses, hipérbolas, parábolas, cônicas, superfícies reguadas, superfícies de revolução, superfícies B-Spline, cilindros tabulados, cilindros, cubos, cones e cunhas ("wedges").

Além disto também permite usar comandos tridimensionais de rotação em torno de um eixo (varredura rotacional) e de projeção ao longo de uma direção (varredura translacional).

Em todos os casos, o usuário deve especificar os parâmetros geométricos e de posição.

CAPÍTULO 5

PADRONIZAÇÃO GRÁFICA

5.1 - INTRODUÇÃO

Este capítulo tem como objetivo analisar a viabilidade da transportabilidade do modelador geométrico desenvolvido, usando outros pacotes gráficos padronizados.

Inicialmente discute-se sobre padronização gráfica, seus fundamentos, história, uma proposta de padrão (CORE) que precedeu aos atuais padrões, e um estudo detalhado de cada um dos padrões atuais.

Os padrões estudados e que aqui são apresentados são: CGI, CGM, GKS, GKS-3D, e PHIGS.

5.2 - FUNDAMENTOS

Suponha que cada fabricante de tomadas elétricas as fizesse com diferentes espaçamentos e os orifícios com diferentes diâmetros, isto implicaria em encontrar no mercado uma série de adaptadores que possibilitaria conectar diferentes aparelhos às tomadas.

Portanto, o único beneficiado seria o fabricante de adaptadores. Um problema análogo seria se cada fabricante de computador ou periférico produzisse os conectores ou os teclados a sua maneira.

A padronização é tão necessária em relação a componentes e equipamentos (hardware) quanto a nível de software. Particularmente, os sistemas CAD/CAM são constituídos de hardware e software.

Os dispositivos gráficos de saída (monitores de vídeo, traçadores gráficos, impressoras, etc.) e de entrada ("mouses", teclados, mesas digitalizadoras, etc.) possuem comandos e protocolos de transmissão de dados diferentes, que variam conforme o tipo, modelo e fabricante.

Logo, o programa que gera uma reta na tela de um monitor de vídeo padrão CGA ("Color Graphics Adapter") é diferente daquele que desenha a mesma reta em um monitor padrão EGA ("Enhanced Graphics Adapter") e também é diferente do programa que gera a mesma reta em um determinado traçador gráfico.

Uma solução é alterar o software, isto é, alterar o programa que gera desenhos em um determinado equipamento de saída criando um novo programa adaptado a outro equipamento.

Porém esta solução foi abandonada com o desenvolvimento do conceito de interface controladora de dispositivo ("device driver interface").

A interface controladora é um conjunto bem definido de rotinas dentro de um programa gráfico que fazem acesso direto ao equipamento. Qualquer acesso a equipamento que se fizer necessário dentro do programa deverá utilizar rotinas desta interface.

Com isso, o custo para migração de software para diferentes equipamentos diminuiu bastante, pois para mudar de equipamento basta reprogramar a interface controladora adaptandoa aos novos comandos do equipamento, sem que o programa do usuário necessite ser mudado.

Apesar desta evolução, a maioria dos sistemas CAD possui suas próprias interfaces controladoras, que tem características similares, mas são incompatíveis entre si, o que obriga ao desenvolvimento de inúmeras controladoras diferentes para um mesmo dispositivo.

Assim, com o objetivo de padronizar todas estas interfaces e evitar a duplicação de esforços, foi criado o conceito de "dispositivo virtual" e o padrão de interface controladora CGI.

O CGI dispõe de funções padronizadas para entrada e saída gráfica, de forma que o programa que as utiliza "enxerga" sempre os mesmos dispositivos virtuais e os aciona sempre da mesma forma, não importando se o dispositivo físico é um monitor de vídeo ou um traçador gráfico, por exemplo. Quem se encarrega de converter os comandos para as sintaxes próprias dos equipamentos é o pacote CGI.

Sistemas escritos com o padrão CGI, possuem a chamada "independência de dispositivo", uma das grandes vantagens da padronização. Da mesma maneira, pacotes gráficos padronizados como o GKS, oferecem funções básicas para o desenvolvimento de sistemas gráficos, tais como geração de elementos geométricos, interação com o usuário, transformações geométricas, geração de bibliotecas de símbolos, etc.

Estas funções, que além de facilitarem o desenvolvimento, tornam os sistemas portáteis, ou seja, estes sistemas podem ser processados em diferentes computadores e estações de trabalho ("workstations") que disponham de tal pacote, sem custo adicional.

A portabilidade é possível de ser obtida em diversos níveis, de acordo com as ferramentas e/ou interfaces padronizadas que se utilize: portabilidade de programas, usando-se pacotes gráficos, linguagens de programação e sistemas operacionais padrões; portabilidade de dispositivos, através de interfaces controladoras padronizadas; e portabilidade de dados, com o uso de arquivos de desenhos ou projetos padronizados.

Entre as desvantagens da padronização, podem-se citar o não acompanhamento do desenvolvimento tecnológico, devido a demora para geração e aprovação de normas, o cerceamento da inovação e da concorrência entre fornecedores de sistemas, e o não aproveitamento de características particulares dos equipamentos visando-se maior eficiência dos programas. Mas as vantagens superam em muito os problemas.

Os cinco atuais padrões gráficos (PHIGS, GKS, GKS-3D, CGI e CGM) são as bases nos quais outros padrões mais especializados irão se basear.

Inicialmente, descreve-se o processo da padronização e um modelo de referência para a computação gráfica.

Segue a descrição do sistema CORE que foi a primeira proposta de pacote de sub-rotinas gráficas padronizadas que deu origem a seu concorrente, o GKS, na disputa de se tornar padrão, e apesar de não ter sido aceito como padrão serviu como base aos outros padrões existentes.

A seguir descreve-se o CGM, um padrão para a transferência de desenhos (padrão para formato de dados gráficos) entre configurações de hardware e software diferentes.

Após o CGM, explica-se CGI, um padrão muito relacionado com CGM, disponível tanto como interface programável padrão ou como padrão de protocolo de dados entre dispositivos.

Por último, descreve-se o GKS e os padrões para definição e visualização de objetos gráficos 3D: o GKS-3D, uma extensão 3D do GKS, e o PHIGS, um padrão que além de suportar todas as funções de visualização do GKS3D, suporta o modelamento de objetos gráficos.

Nestes enfoques é descrita a semântica das funções e não a sintaxe, pois o objetivo nesta comparação apresentada foge da sintaxe dos modelos.

5.3 - O PROCESSO DE PADRONIZAÇÃO

O Instituto Nacional Americano de Normalização ANSI ("American National Standards Institute"), situada em Nova York, é a organização americana que coordena as normas americanas.

ANSI não desenvolve padrões ou normas, ao invés disso, identifica quais padrões são necessários e determina modelos a serem seguidos pelas instituições de normalização com o objetivo de serem padrões de ampla aceitação na indústria. Assim, estas normas resultantes se tornam normas nacionais ou internacionais, disponíveis para a indústria, governos ou público em geral.

ANSI, como membro oficial do governo americano, é membro e participa dos trabalhos na Organização Internacional de Normalização ISO ("International Standards Organization").

Na ISO, o Comitê Técnico 97 ISO/TC97 cuida da normalização no campo da informática. Dentro do ISO/TC97, o Subcomitê 21 SC21, chamado também Subcomitê de Interconexão de Sistema Abertos OSI ("Open Systems Interconnection"), a computação gráfica pertence a seus grupo de trabalho número 2 WG2 ("Working Group 2"). Portanto, a padronização gráfica dentro de ISO é designada como ISO/TC97/SC21/WG2.

Existem outros grupos de trabalho dentro do SC21, tais como: WG1, para a arquitetura OSI e modelos de referência, WG3, para banco de dados, WG5, dedicado a sistemas operacionais, terminais virtuais e transferência de arquivos, etc.

Incluídos no ISO/TC97 como subcomitês separados, no mesmo nível do SC21, estão o SC2 para conjuntos de caracteres e códigos de informação (teletextos, videotextos, codificação de desenhos e imagens, codificação de audio e "facsmile"); o SC22 para as linguagens de programação; etc.

Em ISO/TC97, um novo projeto começa quando um subcomitê, como o SC21, ou um membro de um país, como ANSI, elabora um Novo Item de Trabalho NWI ("New Work Item") e o submete ao TC97 para votação. Cada membro de um país tem um voto.

Por exemplo, ANSI encaminhou os NWI e os seus respectivos documentos, ao ISO/TC97 para as propostas de CGM, CGI e PHIGS.

A partir de um NWI, o SC21/WG2 prepara um Projeto de Trabalho WD ("Working Draft"), o qual quando aceito e completo, por recomendação do SC21/WG2 ao SC21, pode ser registrado como uma Proposta de Projeto DP ("Draft Proposal"). Se o WD é aceito entre os membros do SC21 como uma DP, ISO atribui um número à norma proposta, por exemplo, GKS é ISO 7942.

Após o DP ter um consenso geral (ser considerado tecnicamente estável), é redigido e posto em votação entre o TC97 e todos os membros dos países pertencentes a ISO. Este documento recebe o nome de Projeto Padrão Internacional DIS ("Draft International Standard") acompanhado do número respectivo, por exemplo, GKS foi DIS 7942.

Se o novo DIS é aprovado por ISO, o DIS é publicado anexando-lhe o ano de publicação, por exemplo, GKS é conhecido como ISO 7942-1985.

O Brasil através da ABNT (Associação Brasileira de Normas Técnicas) participa como membro votante no ISO/SC21 desde 1986 com os trabalhos da Comissão de Computação Gráfica do CB-21 (Comitê de Informática) da ABNT.

A norma GKS brasileira já foi produzida e está indo para votação. Assim mesmo já existem grupos de trabalho para a elaboração das normas CGM, CGI, PHIGS e outros estudos nas áreas de processamento de imagens, interação homem-máquina, validação e testes, e terminologia de computação gráfica.

5.4 - MODELO DE REFERÊNCIA PARA A COMPUTAÇÃO GRÁFICA

5.4.1 - INTRODUÇÃO

O Comitê de Padronização Gráfica ISO/TC97/SC21/WG2, apresenta um modelo básico de referência para a computação gráfica.

Este modelo é baseado nos modelos implícitos contidos nos padrões propostos como GKS, PHIGS, CGM e CGI, e serve como base para coordenar as normas de computação gráfica em desenvolvimento, e para definir as interfaces entre as normas da computação gráfica e aquelas nas outras áreas do campo da informática.

Os modelos conceituais surgiram com as normas gráficas desde os primeiros esforços de padronização.

A partir dos modelos conceituais, nasceu a proposta de separar a exibição de gráficos do modelamento de objetos gráficos, assim como separar um sistema gráfico em partes dependentes de dispositivo e independentes de dispositivo.

Os primeiros esforços contudo não produziram modelos integrados, pois técnicas gráficas não eram utilizadas tão extensamente como são atualmente. A padronização parecia ser adequadamente servida pelos modelos informais e implícitos.

Desde então os gráficos tem se tornado uma tecnologia barata e penetrante usados pela maioria das aplicações computacionais.

A rápida evolução da tecnologia no processamento distribuído e em redes de computadores, criou um potencial para novas interfaces dentro dos sistemas gráficos. Uma série de normas potencialmente relacionadas com computação gráfica foram aprovadas ou estão em desenvolvimento.

Portanto, estes eventos demonstram que modelos de referência explícitos e formais são necessários em computação gráfica para coordenar o desenvolvimento de múltiplas normas gráficas e permitir o desenvolvimento independente mas coordenado de normas relacionadas com gráficos.

5.4.2 - O MODELO DE REFERÊNCIA

O propósito do modelo de referência é servir como base para a padronização.

Deve descrever completamente (em um dado nível de abstração) uma área particular a ser padronizada, e servir como identificador de normas necessárias em outras áreas, e de relações que devem existir entre estas normas.

Também, deve fornecer uma referência comum para manter a consistência de todas as normas relacionadas.

Apesar disto, o modelo não é por si mesmo um padrão, nem serve como uma especificação ou meio de avaliação de uma implementação, nem proporciona um nível suficiente de detalhes para definir precisamente as funções e interfaces da arquitetura gráfica, apenas é uma estrutura conceitual e funcional.

Existe uma diferença entre a estrutura de normas e a arquitetura de sistemas reais. As normas em geral servem para especificar detalhes externamente visíveis de itens sendo padronizados e não os detalhes internos de uma implementação em particular.

5.4.2.1 - VISÃO DE ALTO NÍVEL DO MODELO DE REFERÊNCIA

A figura (Fig. 5.1) a seguir, mostra o ambiente da computação gráfica dentro dos sistemas de processamento de informação.

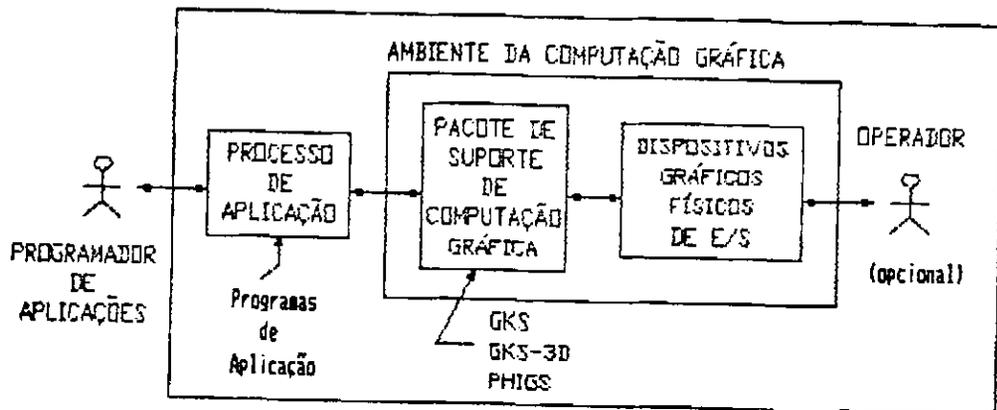


Fig. 5.1 - O ambiente da computação gráfica.

Programas de aplicação interagem com um ou mais dispositivos gráficos físicos através de pacotes de suporte.

O termo "pacote de suporte de computação gráfica", representa o nível de funcionalidade fornecido por padrões como GKS, GKS-3D e PHIGS.

O operador interage com um programa de aplicação observando um ou mais dispositivos de saída e manipulando um ou mais dispositivos de entrada.

A função do pacote de suporte, é aliviar ao programador de aplicações de envolver-se com a idiossincrasia de particulares dispositivos físicos, proporcionando uma interface de alto nível para sua funcionalidade.

5.4.2.2 - VISÃO DE BAIXO NÍVEL DO MODELO DE REFERÊNCIA

A figura (Fig. 5.2) abaixo, ilustra alguns dos conceitos usados para modelar o ambiente da computação gráfica.

Estes conceitos são baseados em princípios do sistema CORE (que originou o GKS) e do GKS, porém não são amplamente entendidos fora da comunidade das normas gráficas.

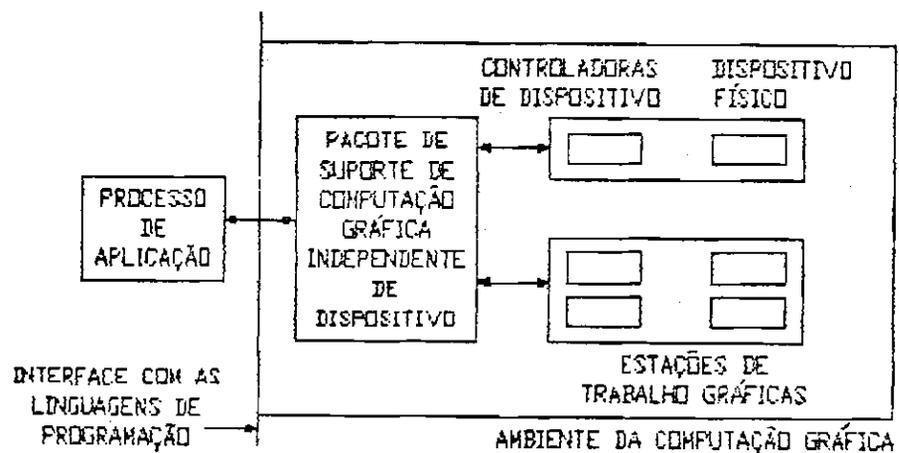


Fig. 5.2 - Ambiente da computação gráfica em um nível mais baixo.

A figura (Fig. 5.3) a seguir, mostra o modelo de referência para a computação gráfica e os padrões gráficos atuais.

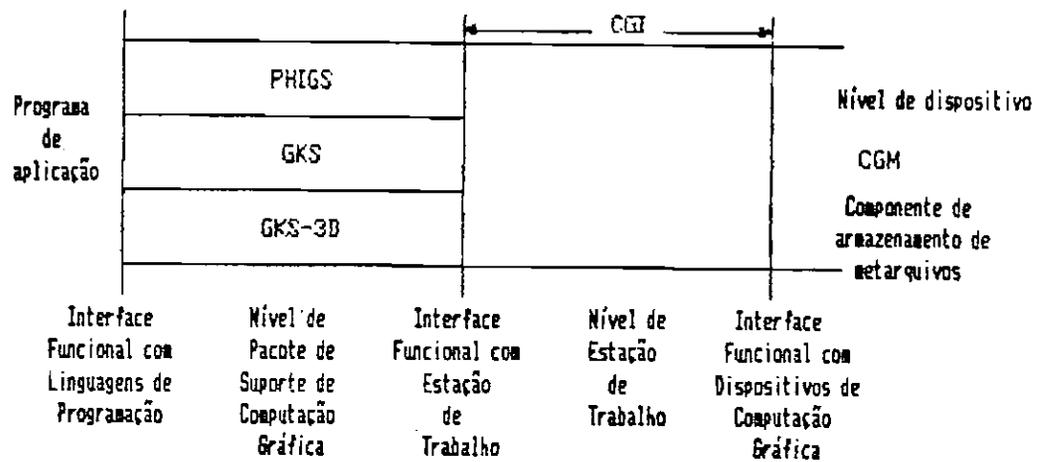


Fig. 5.3 - Modelo de referência da computação gráfica e padrões gráficos.

O modelo de referência, como mostra a figura (Fig. 5.3) acima, é constituído dos seguintes componentes e interfaces:

1. Programa de aplicação
2. Interface funcional com linguagens de programação
3. Nível de pacote de suporte de computação gráfica
4. Interface funcional com estação de trabalho
5. Nível de estação de trabalho
6. Interface funcional com dispositivo de computação gráfica
7. Nível de dispositivo

8. Componente armazenador de metarquivos ("metafile")

A interface de mais alto nível é a interface funcional com as linguagens de programação sendo desenvolvida para a padronização gráfica, a qual define a fronteira entre um programa de aplicação e o pacote de suporte de computação gráfica.

Dentro do pacote de suporte de computação gráfica se encontra a maior parte de software gráfico independente de dispositivo em tais sistemas tais como, GKS, GKS3D e PHIGS. Todas as transformações independentes da estação de trabalho e os elementos de armazenamento se encontram aqui.

Entre o nível de pacote de suporte de computação gráfica e o nível de estação de trabalho, se encontra a interface funcional com a estação de trabalho. Esta, é uma interface abstracta desde que ainda discute-se sobre a sua necessidade.

É importante ressaltar que o termo "estação de trabalho" dentro da padronização gráfica, é entendido como uma coleção bem definida de dispositivos de entrada e saída, enquanto que o uso comun deste termo se refere a um poderoso computador pessoal.

O nível de estação de trabalho, inclui funções específicas a uma única estação de trabalho e não a um dispositivo gráfico em particular.

Por exemplo, elementos de armazenamento dependentes da estação de trabalho, ou transformações de estação de trabalho, que podem, ou não, ser armazenados em um dispositivo gráfico.

A interface funcional com dispositivo de computação gráfica, é a interface de mais baixo nível. Esta interface é o limiar entre a estação de trabalho e um único dispositivo gráfico de entrada ou saída. Em geral, varia para cada dispositivo gráfico em função da inteligência do dispositivo.

Abaixo da interface funcional com dispositivo de computação gráfica se encontra o nível de dispositivo. As funções dependentes de dispositivo dentro da arquitetura gráfica e os componentes de armazenamento de "metarquivos" (arquivos de desenho) de encontram aqui.

Os padrões gráficos existentes e os propostos, tais como GKS, GKS-3D e PHIGS, se encontram no nível de pacote de suporte de computação gráfica, e suas ligações com suas linguagens de programação, são as interfaces funcionais de linguagens de programação.

CGM, representa a interface com um componente capaz de armazenar metarquivos, no nível da interface funcional com dispositivo de computação gráfica.

CGI, representa a família de interfaces em um nível dentro do modelo de referência abaixo do pacote de suporte de computação gráfica e acima do nível de dispositivo.

5.5 - O SISTEMA CORE

5.5.1 - INTRODUÇÃO

O sistema CORE é um pacote de sub-rotinas gráficas padronizadas desenvolvido pelo Comitê de Planejamento de Normas Gráficas GSPC ("Graphics Standards Planning Committee") do Grupo de Interesse Especial por Gráficos SIGGRAPH ("Special Interest Group for Graphics") da Associação para Máquinas Computadorizadas ACM ("Association for Computing Machinery").

A definição inicial de CORE foi publicada em 1977 e uma revisão atualizada com extensões para gráficos "raster" em 1979.

A capacidade funcional de CORE divide-se em:

1. Primitivas de saída

São primitivas tanto 2D como 3D, usadas para definir os objetos a serem mostrados. As primitivas, as quais são linhas, cadeias de caracteres, polígonos e marcadores, são especificados no sistema de coordenadas da aplicação, ou do mundo, WCS ("World Coordinate System").

2. Atributos das primitivas de saída

Tais como cor de linha, tipo de linha, intensidade, fonte de caracteres (estilo), índice de interior de polígono, índice de borda de polígono, etc.

3. Segmentos

Grupos de diferentes primitivas de saída. Cada grupo forma uma unidade modificável nas aplicações interativas, e pode ser criado, apagado, transformado, realçado ("blinked"), tornado temporariamente invisível ou selecionado usando um dispositivo de entrada.

4. Capacidade de visualização

Para criar a imagem de um objeto em uma vista ("viewport") dentro de uma superfície de exibição.

Qualquer projeção geométrica plana pode ser usada para a visualização. Transforma-se uma área retangular ou janela ("window"), usada para delimitar a imagem no sistema de coordenadas do mundo, neste segundo retângulo ou vista na superfície de exibição do dispositivo de saída.

5. Interação com operador

A interação é feita através de dispositivos de entrada tais como identificador lógico ("logical pick"), localizador ("locator"), selecionador ("button" ou "choice"), cadeia de caracteres ("text"), cadeia de posições ("stroke") e avaliador ("valuator").

Os protótipos físicos destes dispositivos são caneta de luz ("light pen"), ratinho ("mouse") e manche, chaves seletoras ou teclas de função de um localizador, teclado alfanumérico e de funções programáveis, localizadores mais mesa digitalizadora ("tablet"), e teclado ou potenciômetros, respectivamente.

6. Controle geral

Para habilitar/desabilitar os dispositivos de entrada, selecionar a superfície de visualização na qual objetos serão visualizados, estabelecer atributos "default" e habilitar/desabilitar o recorte ("clipping").

5.5.2 - INTERFACE DI/DD (INDEPENDENTE / DEPENDENTE DE DISPOSITIVO)

O sistema CORE assim como outros pacotes independentes de dispositivo, define um conjunto de funções e as relaciona com o hardware real em dois passos: o primeiro é "independente de dispositivo" e o segundo é "dependente de dispositivo".

Como exemplo, na fase independente de dispositivo, a transformação de visualização é utilizada para transformar uma linha (primitiva de saída) das coordenadas do mundo WCS (geralmente coordenadas cartesianas) em coordenadas de dispositivo normalizado NDC ("Normalized Device Coordinates"), a qual é uma superfície de exibição virtual representado pelas coordenadas (0,0) a (0,1) onde x e y variam de 0 a 1.

Na fase dependente de dispositivo, as coordenadas NDC são transformadas em coordenadas de dispositivo DC ("Device Coordinates") e comandos do dispositivo são usados para mostrar a linha.

A implementação CORE consiste de uma única parte, independente de dispositivo (pacote de funções gráficas), mais uma controladora ("driver") dependente de dispositivo, para cada dispositivo gráfico diferente, como mostra a figura (Fig. 5.4) abaixo:

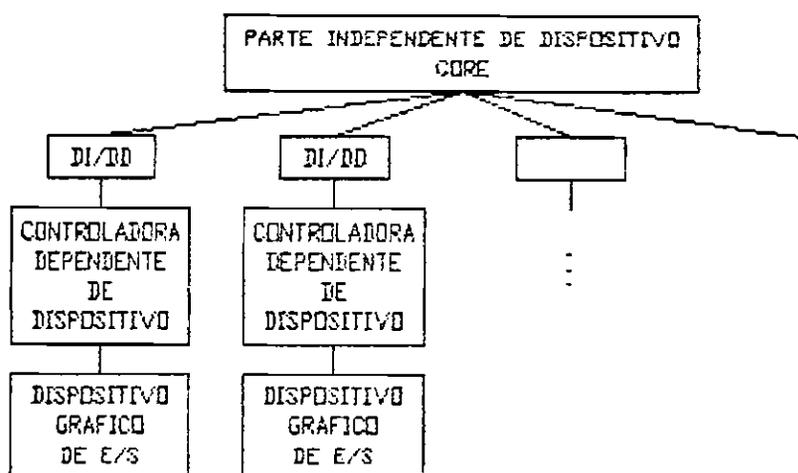


Fig. 5.4 - Interface independente/dependente de dispositivo DI/DD.

Assim para suportar um novo dispositivo gráfico basta escrever uma nova controladora dependente de dispositivo.

O projeto da interface DI/DD define uma "estação gráfica virtual", e entre os seus objetivos, podem ser citados:

1. Funcionalidade completa

Com o objetivo de aproveitar ao máximo as vantagens dos dispositivos de visualização.

A interface aceita primitivas de saída transformadas no espaço NDC em 3D, suporta a maioria dos atributos destas primitivas, pode criar segmentos e modificar seus atributos, e proporciona todos os dispositivos de entrada lógica.

Não executa transformações de visualização, não mantém fila de eventos, nem checagem de erros.

Assim, com esta interface de alto nível, facilita-se a tarefa de escrever controladoras para dispositivos simples, tais como "plotters" e "displays raster", se todas as funções necessárias estão embutidas dentro da controladora de dispositivo.

Um arquivo de simulação PDF ("pseudo-display file") é mantido para armazenar temporariamente os segmentos. Os atributos das primitivas de saída, assim como os atributos dos segmentos, não suportados pelo hardware são simulados.

Ao inicializar a controladora, esta pode reportar quais funções suporta, permitindo que a parte do sistema CORE independente de dispositivo simule estas necessidades, não fornecidas pela controladora, simplificando a implementação da controladora.

2. Visível divisão entre parte independente e parte dependente de dispositivo.

Com o objetivo de manter o mecanismo da interface DI/DD o mais simples possível com uma clara diferença entre as duas partes.

Isto, para que a controladora possa, de fato, rodar em um processador separado, de modo que nenhuma variável seja compartilhada através da interface.

5.5.3 - ESTRUTURA DA SAÍDA - PARTE CORE INDEPENDENTE DE DISPOSITIVO

O projeto da estrutura de saída do sistema CORE é composto pelo processo de visualização ("viewing pipeline"), o transmissor ("dispatcher"), o sistema de arquivos PDF e as simulações.

Esta estrutura está mostrada na figura (Fig. 5.5) a seguir:

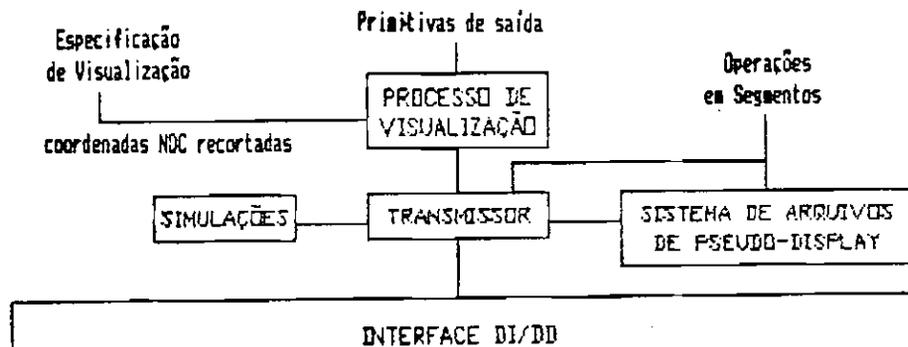


Fig. 5.5 - Estrutura da saída da parte CORE independente de dispositivo.

5.5.3.1 - PROCESSO DE VISUALIZAÇÃO

O processo de visualização aplica a transformação de visualização nas primitivas de saída para serem recortadas, projetadas ou convertidas em coordenadas NDC.

Entre os parâmetros que definem a transformação de visualização, estão a normal do plano da vista, o ponto de referência da vista, o centro da projeção, a janela, e a vista.

A transformação de visualização corrente é aplicada no segmento sendo criado, e a definição desta transformação corrente pode ser alterada.

A saída do processo de visualização é colocada em um "pacote de comunicação" (um vetor), o qual é passado ao transmissor. Existem formatos de pacote para cada tipo de primitiva de saída: "marker", texto, linhas, "polymarkers", polilinha e polígono.

5.5.3.2 - TRANSMISSOR

O transmissor controla o subsistema de saída como mostra a figura (Fig. 5.5) anterior.

Decide quais pacotes de informação devem ser enviados através da interface DI/DD à controladora de dispositivo atualmente selecionada, e quais pacotes devem ser armazenados no sistema PDF, para posterior uso, todas as vezes que uma nova ação de visualização ("refresh") seja requisitada.

Baseado nos atributos das primitivas de saída e nas capacidades do dispositivo, reportadas pela controladora (quando a superfície de visualização foi selecionada), o transmissor também decide se uma simulação é necessária e chama as rotinas de simulação apropriadas.

Os pacotes de informação armazenados no PDF passam direto para a controladora de dispositivo através da interface DI/DD quando são solicitados pelo subsistema de entrada (explicado mais a frente) para uma nova ação de "refresh".

Várias controladoras de dispositivo podem ser associadas (linkadas) ao sistema CORE independente de dispositivo, porém as primitivas de saída são enviadas na superfície de visualização do dispositivo atualmente selecionado. Para evitar, a sobrecarga de controladoras na memória, na linkagem pode ser escolhida qual, ou quais, controladoras serão usadas.

Assim, uma aplicação pode usar diferentes superfícies de visualização em diferentes tempos. Todas as controladoras de dispositivo são carregadas para permitir, em tempo de execução, a seleção de uma determinada superfície de visualização. A cada superfície de visualização é atribuído um identificador numérico.

5.5.3.3 - ARQUIVO DE "PSEUDO-DISPLAY" PDF

Um arquivo PDF é mantido para os dispositivos de visualização. Os dispositivos de visualização se reportam ao sistema CORE independente de dispositivo, indicando que o buffer de visualização ("display buffer") está vazio (não existe arquivo de visualização).

A estrutura lógica do PDF é um conjunto de segmentos e seus atributos. Cada segmento é uma sequência de primitivas de saída e seus respectivos atributos. A estrutura física do PDF é formada por dois vetores. Um vetor armazena os nomes dos segmentos e seus atributos, e o outro, armazena a sequência de primitivas de saída e seus atributos.

Pacotes de informação, no mesmo formato como são passados ao transmissor, são enviados ao sistema PDF pelo transmissor, para poderem ser usados quando solicitada uma nova ação de visualização.

5.5.3.4 - SIMULAÇÕES

Com o objetivo de manter a independência de dispositivo, proporciona-se a simulação de certos atributos das primitivas de saída: tipo de linha, largura de linha, preenchimento de polígonos, tipo de lados de um polígono, e a intensidade em um dispositivo colorido usado como dispositivo monocromático.

Os atributos de segmentos como o realce e as transformações da imagem, também podem ser simuladas.

As rotinas de simulação são invocadas pelo transmissor antes das primitivas serem enviadas através da interface DI/DD às controladoras de dispositivo.

Os pacotes de informação podem vir através do transmissor de duas fontes: do processo de visualização ou do sistema PDF. O transmissor decide quais atributos devem ser simulados na superfície de visualização atualmente selecionada pela controladora.

5.5.4 - ESTRUTURA DA ENTRADA - PARTE CORE INDEPENDENTE DE DISPOSITIVO

A entrada da parte independente de dispositivo do sistema CORE, é composta pela fila de eventos ("event queue"), manipulação ("handling") de eventos síncronos e assíncronos, manipulação de associações e eco, e o uso do arquivo de pseudo-display PDF.

A figura (Fig. 5.6) abaixo, mostra esta estrutura:

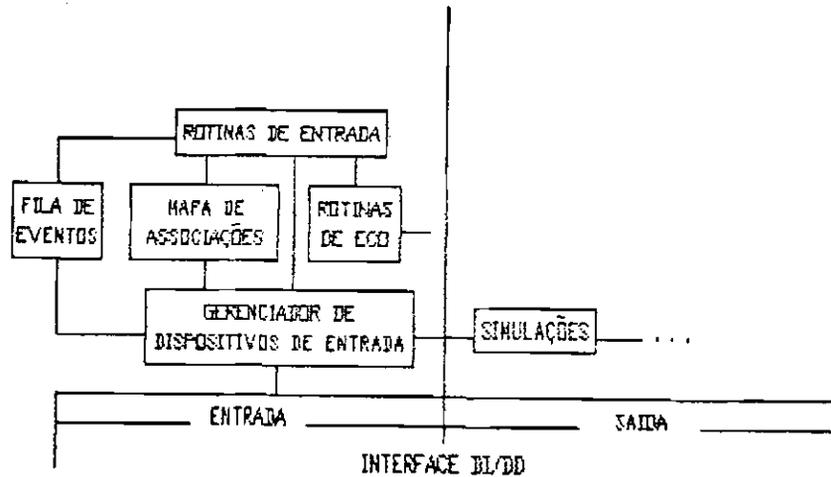


Fig. 5.6 - Estrutura de entrada da parte CORE independente de dispositivo.

5.5.4.1 - FILA DE EVENTOS

A fila de eventos é mantida pelo sistema CORE, como um vetor que representa um conjunto de descrições de eventos ("event reports") devolvidos pelas controladoras de dispositivo, incluindo informações de quaisquer dispositivos amostrados associados.

5.5.4.2 - ASSOCIAÇÕES

As associações existem para linkar dispositivos amostrados com eventos de dispositivos. Como exemplo, um "mouse" (ratinho) pode ser associado com um certo botão. Quando este botão é apertado, o "mouse" é lido e seu valor (0 não apertado e 1 apertado) é incluído junto com a descrição ("report") do evento de botão.

O mapa de associação, mostrado na figura (Fig. 5.3) anterior, contém uma entrada para cada evento de dispositivo para indicar qual dispositivo amostrado está associado com este evento.

5.5.4.3 - ENTRADA SÍNCRONA/ASSÍNCRONA

A entrada do sistema CORE pode ser implementada em modo "síncrono" ou "assíncrono", dependendo da capacidade da controladora de dispositivo de entrada. Para o programa de aplicação, usando o sistema CORE, o tipo de implementação usado é transparente.

Uma controladora de dispositivo, que opera em modo síncrono, retorna dados de evento somente quando uma rotina do tipo ESPERA_EVENTO é chamada.

Após a chamada, a controladora inicia um "loop" de espera ("polling") até receber resposta de qualquer um dos dispositivos de evento habilitados, para logo retornar a descrição do evento ("event report") ao sistema CORE, o qual o torna a descrição de evento atual, e retorna ao programa de aplicação. Neste modo, a fila de eventos sempre está vazia.

Por outro lado, uma controladora de dispositivo assíncrona, tem um manipulador de interrupções ("interrupt handler"), que coloca eventos na fila de eventos, conforme estes vão acontecendo.

Um evento ocorre quando um dos vários dispositivos de evento é habilitado. O manipulador de interrupções aceita o evento e amostra seu dispositivo associado. A fila de eventos logo é bloqueada de tal maneira que as rotinas independentes de dispositivo não possam acessá-la, a descrição do evento é colocada na fila de eventos, e a fila é liberada.

Seja qual for o modo de operação, o sistema CORE retorna para o programa de aplicação o primeiro evento da fila de eventos, como sendo a descrição de evento atual. O programa de aplicação, pode então chamar as rotinas apropriadas para obter os dados da descrição do evento.

5.5.5 - INTERAÇÃO ENTRE ENTRADA E SAÍDA

Existem várias interações entre os subsistemas de entrada e saída do sistema CORE. Como exemplo, pode ser citado o eco. Na edição, ou seleção do valor de entrada pelo operador, o subsistema de entrada gera uma saída (eco) através do transmissor para informar na tela de entrada o valor corrente da entrada.

5.5.6 - TRATAMENTO DE ERROS

Todos os erros são reportados ao usuário de uma forma padronizada. O nome da sub-rotina chamada, os parâmetros da chamada, e um código de erro, são passados a um módulo de reportagem de erros. O nome da sub-rotina, seus parâmetros, o número do erro e a mensagem do respectivo erro (extraída de um arquivo de mensagens de erro), são mostrados na superfície do dispositivo selecionado pelo usuário.

5.5.7 - ESTRUTURA DEPENDENTE DE DISPOSITIVO

As controladoras de dispositivo possuem um módulo de alto nível o qual é essencialmente uma declaração de controle de múltipla escolha (declaração "case" da linguagem C), que analisa o código da função de entrada, ou saída, passado à controladora como parte de seus argumentos. Qualquer função que não é suportada pela controladora é ignorada por esta declaração de controle.

Devido ao sistema CORE independente de dispositivo, ser responsável pelo fornecimento de simulação de funções que a controladora de dispositivo não suporta, a controladora de saída é geralmente simples.

Uma controladora "bufferizada" mantém uma tabela de segmentos e uma lista de dispositivos de visualização.

Informação de "status" é retornada da controladora de dispositivo no pacote de comunicação. Além disto, um código de erro é retornado se qualquer condição que não pode ser resolvida, como erro de hardware ou estouro de tabela, é detectado na controladora de dispositivo.

As controladoras de dispositivo mantêm suas próprias tabelas de estado independente uma da outra, e do sistema CORE independente de dispositivo.

5.6 - CGM - "COMPUTER GRAPHICS METAFILE"

5.6.1 - INTRODUÇÃO

O CGM é o único padrão para especificação de banco de dados gráfico. É um padrão complexo para a captura de múltiplas definições de desenhos independentes de dispositivo.

O CGM é a primeira norma para a definição de um metarquivo gráfico ("graphical metafile") de propósitos gerais. Este metarquivo gráfico, é um arquivo para a captura, transferência e armazenamento de informação gráfica.

5.6.2 - DEFINIÇÃO DE METARQUIVO GRÁFICO

Considere duas aplicações gráficas em diferentes áreas locais de uma rede de computadores.

Em um área, um programa de aplicação gera gráficos de dados técnicos os quais são salvos para posterior impressão. O formato do arquivo salvo é binário, semelhante a um protocolo de baixo nível entre dois dispositivos gráficos, mas que não corresponde a nenhum dispositivo gráfico existente.

Em outra área da rede, um usuário gera interativamente desenhos usando um pacote gráfico baseado em GKS. Durante a sessão, um texto do diálogo em formato ASCII, através da interface GKS da estação de trabalho, é armazenado para posterior edição.

Apesar dos dois arquivos gráficos produzidos por estas duas aplicações hipotéticas pareçam distintas, ambos são metarquivos gráficos, pois o conceito de metarquivo gráfico é ampla: um metarquivo gráfico é um mecanismo para a captura, armazenamento e transporte de informação gráfica.

Os metarquivos gráficos permitem:

1. Formatos de dados para armazenamento de desenhos
2. Protocolos gráficos para impressão "off-line" (fora de linha) e "off-site" (fora de estação)
3. Formatos únicos para redirecionamento ("spooling") a múltiplos e diferentes dispositivos de impressão
4. A possibilidade de ter uma única interface padrão para dispositivos geradores de gráficos
5. O reuso do mesmo desenho sem necessidade de um novo processamento do mesmo
6. Mecanismos de armazenamento e recuperação de desenhos
7. Unificar e integrar diferentes aplicações gráficas e sistemas de hardware/software em sistemas distribuídos

5.6.3 - RELAÇÕES DE UM METARQUIVO COM UM SISTEMA GRÁFICO

Um metarquivo é um banco de dados gráfico. Portanto, deve existir um componente dentro do sistema gráfico para a geração do banco de dados em paralelo com a execução de uma aplicação, este componente é o "gerador de metarquivos".

Também, deve existir outro componente para a leitura, interpretação e tradução da informação gráfica de um metarquivo, este componente é o "interpretador de metarquivo".

A figura (Fig. 5.7) abaixo, ilustra a relação entre um metarquivo CGM, e os dois componentes mencionados acima, com o resto do sistema gráfico.

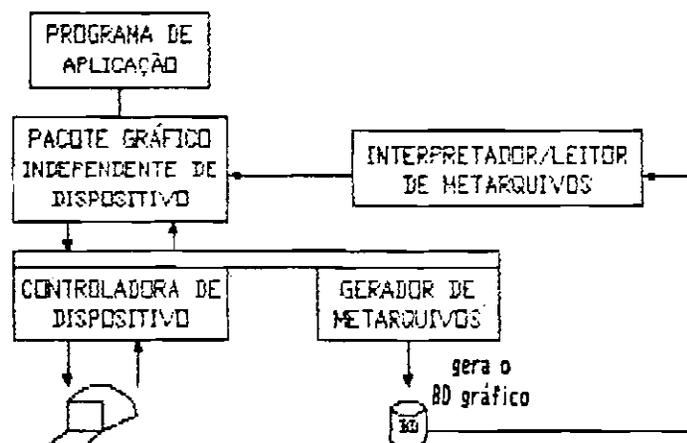


Fig. 5.7 - A relação de CGM com um ambiente de aplicações gráficas.

O gerador e o interpretador são componentes de software, porém, principalmente as funções do interpretador, podem ser migradas para hardware. Esta migração, é de fato um dos propósitos e benefícios antecipados da padronização de metarquivos.

O que fica implícito, é a separação do processo gerando metarquivos e o processo usando estes metarquivos. Apenas existe um único caminho em uma única direção que conecta ambos processos.

Esta limitação representa implicações no projeto de metarquivos e na seleção dos elementos do metarquivo, e é uma das principais diferenças entre um metarquivo e uma interface como CGI. Em particular, as funções de um metarquivo devem ser independentes do dispositivo final de saída.

5.6.4 - TIPOS DE METARQUIVOS

Para entender o que CGM é e não é, deve-se inicialmente distinguir dois tipos de metarquivos: captura de desenho e captura de sessão. Esta classificação, não é restrita; pois a definição de um metarquivo pode conter características de ambos tipos.

Um metarquivo de captura de desenho, é aquele cuja função primária é a captura de múltiplas definições de desenhos independentes de dispositivo. É um mecanismo para armazenar, ou transmitir, randomicamente, coleções de desenhos independentes. CGM é um metarquivo de captura de desenhos.

Um metarquivo de captura de sessão, é designado para capturar o diálogo completo de saída através de alguma interface em um sistema gráfico.

É um mecanismo para a gravação do estado exato de um sistema gráfico durante uma sessão gráfica. O metarquivo GKS (GKSM) anexo ao GKS contém tal definição, porém não faz parte da norma.

5.6.5 - DEFINIÇÃO DE CGM

CGM é um metarquivo estático de captura de desenhos, isto é, não contém elementos (funções) com efeitos dinâmicos em desenhos parcialmente definidos. Uma mudança de transformação para obter "zoom", é um exemplo de efeito dinâmico.

A definição do escopo de CGM, particularmente sua limitação, foi discutida durante sua especificação. Uma maioria desejava um metarquivo de captura de imagem que pudesse ser padronizado rapidamente, enquanto outros, especialmente usuários de GKSM, desejavam características dinâmicas, captura de sessão, segmentação e outra técnicas avançadas.

O escopo limitado foi adotado, deixando as características avançadas para uma segunda fase de padronização na qual a prioridade seria estender CGM para servir plenamente ao GKSM.

CGM não é uma padrão de programação de aplicações como GKS e PHIGS, ao contrário, é um padrão de especificação para projetistas de sistemas. A figura (Fig. 5.7) anterior, mostra que a posição conceitual do gerador de metarquivos, está quase no nível das controladoras de dispositivos.

CGM foi projetado para se usado com uma ampla variedade de dispositivos, aplicações e sistemas. O mesmo metarquivo, pode ser interpretado em um terminal monocromático de baixa resolução, em um "plotter" de alta resolução com múltiplas penas, ou em um dispositivo "raster" com alta funcionalidade.

O gerador, pode direcionar o metarquivo a um dispositivo particular, ou a classes de dispositivos, e recortar o conteúdo do metarquivo de acordo com o dispositivo, como por exemplo, o recorte do sistema de coordenadas do metarquivo. O mecanismo de recorte preserva a independência de dispositivo do metarquivo resultante.

5.6.6 - O QUE CGM PADRONIZA

CGM padroniza a semântica e sintaxe de um conjunto de elementos para uma definição de desenhos independentes de dispositivo. A padronização está organizada em 4 partes:

1. Especificação funcional.

Todos os elementos normalizados são identificados, seus parâmetros são descritos e seus significados definidos.

2. Codificação por caracter

Os dados são codificados com caracteres do conjunto ASCII, resultando em caracteres imprimíveis. A codificação é compacta, e pode ser transmitida diretamente através de serviços padronizados de comunicação orientados a caracteres.

Não existem caracteres de controle e fuga, para não confundir os serviços de comunicação.

3. Codificação binária

Especial para aplicações onde a velocidade de geração, e a velocidade de transmissão, são mais importantes. Também é razoavelmente compacta.

4. Codificação texto ("cleartext")

É humanamente legível. Por exemplo, um círculo centrado em (0,0) com raio 2.5 poderia ser codificado como CIRCLE 0,0,2.5.

É transferível com serviços padrões orientados a caracteres (codificação por caracteres), mas não é muito compacta e é relativamente lenta para ser gerada e interpretada. Uma característica importante, é sua facilidade de compreensão e manipulação usando editores de texto.

As partes 2, 3 e 4, representam a codificação de dados da funcionalidade dos elementos CGM da parte 1. Estes 3 tipos de codificação diferentes, são usados para atender as necessidades conflitantes das aplicações, tais como: compacidade do metarquivo versus velocidade de geração, interpretação versus legibilidade, edição, facilidade de transferência, etc.

A capacidade de "tradução" de CGM, permite que qualquer metarquivo CGM em um determinado código, possa ser traduzido a um metarquivo "equivalênte" nas duas outras codificações.

Equivalência, significa que nenhuma informação é perdida e que a interpretação dos metarquivos conduzirá ao mesmo desenho. Portanto, a tradução de um código para outro, e de volta ao primeiro, deve produzir o mesmo desenho, apesar da codificação ser diferente.

Embora se padronize a codificação dos elementos de CGM, nada se especifica sobre os formatos físicos dos registros dos dados codificados. Estes formatos são importantes para um intercâmbio bem sucedido entre metarquivos, porém esta especificação foge do escopo dos comitês de padronização gráfica, por ser do domínio dos grupos de normalização de estrutura, transferência e organização de arquivos.

5.6.7 - ESTRUTURA DE CGM

Um metarquivo de computação gráfica, é uma sequência ordenada de elementos com uma estrutura simples de dois níveis, como mostra a figura (Fig. 5.8) a seguir:

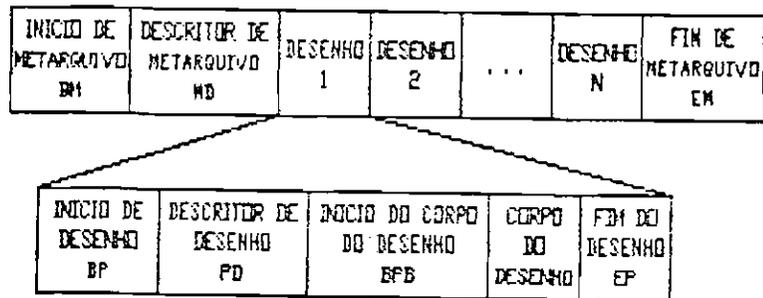


Fig. 5.8 - Estrutura de CGM.

Cada metarquivo consiste de um descritor de metarquivo MD ("metafile descriptor") e uma coleção de desenhos logicamente independentes. Cada desenho, consiste de um descritor de desenho PD ("picture descriptor") e um corpo do desenho contendo a atual definição do desenho.

O MD contém informação que descreve todos os desenhos no metarquivo. Esta informação, permite ao interpretador analisar o metarquivo e identificar os recursos necessários para traduzir o desenho corretamente. O PD também contém informação descritiva, porém só do desenho no qual o PD reside.

Cada definição de desenho é logicamente independente de todas as outras definições de desenho no metarquivo. Após o MD ser interpretado, os desenhos podem ser acessados randomicamente e interpretados corretamente, sem a necessidade de interpretar algum desenho que os anteceda. Esta independência de desenhos, foi um dos dois mais significantes critérios de projeto de CGM.

As coordenadas dos elementos de CGM são chamadas de coordenadas de dispositivo virtual VDC ("Virtual Device Coordinates"). O espaço VDC, é um espaço de coordenadas cartesianas bidimensional. As primitivas gráficas que formam os elementos do CGM, definem os objetos geométricos que formam o desenho. Estas primitivas são as mesmas primitivas de saída do GKS (seção 5.8.4).

A seleção de cores é suportada por CGM em 2 modos: no "modo indexado", o especificador de cor é um índice na "tabela de cores" da estação de trabalho, no "modo direto", o especificador de cor é uma tripla RGB.

RGB, é o único sistema de cores suportada por CGM. Outros sistemas, tais como HLS ("Hue, Lightness, Saturation"), são mais amigáveis ao usuário, mas CGM não é uma norma a nível de usuário e outros sistemas são facilmente convertidos a especificações RGB equivalentes.

5.6.8 - APLICAÇÕES DE CGM

CGM pode ser utilizado de várias maneiras, tanto em sistemas de computação gráfica centralizados ou distribuídos. Três exemplos são dados de como CGM pode ser aplicado a (e recortados para) tipos particulares de uso:

1. Acesso a dispositivos gráficos via um sistema de redirecionamento ("spooling")
2. Armazenamento de desenhos gerados por computador
3. Descrição de páginas, contendo texto e gráficos misturados

Para cada uma destas aplicações serão descritas as vantagens de usar CGM, a relação entre o sistema gráfico e CGM, e as funções de CGM que são especialmente úteis para a aplicação.

5.6.8.1 - ACESSO DE DISPOSITIVOS GRÁFICOS REDIRECIONADOS

Um dos usos mais comuns de qualquer metarquivo gráfico, é transferir informação de um programa de computador para um dispositivo gráfico que é muito lento, está muito longe, ou muito ocupado, para atender o desenho solicitado, em relação ao tempo usado pelo programa para gerar o desenho. Nestes casos, como antigamente, para a saída das impressoras, um sistema de redirecionamento ("spooling") é usado.

Adotar CGM como um formato aceitável para o sistema de redirecionamento é vantajoso por uma série de razões:

- se a codificação de caracteres é usada, o formato é bastante compacto comparado com a maioria das usuais codificações de fabricantes;
- se a codificação binária é usada, o esforço computacional necessário para a geração e interpretação pode ser minimizado, e a codificação é ainda bastante compacta;
- a ampla variedade de funções gráficas disponíveis em CGM permite maior compressão de dados;

- os dados são transportáveis através das redes que aceitam apenas entidades de 7 bits (assumindo que a codificação apropriada é usada);
- desde que o arquivo não precisa conter informação específica para um determinado dispositivo, a saída pode ser redirecionada a outros dispositivos gráficos, se o dispositivo original foi incorretamente especificado, ou está fora de serviço.

5.6.8.2 - ARMAZENAMENTO DE DESENHOS

Em várias aplicações, desenhos gráficos são armazenados em períodos que variam de minutos a anos. O menor tempo, é tipicamente encontrado em saída gráfica dentro de um programa tarefa ("batch"). O tempo intermediário, é encontrado quando se mantém informação "on line" que pode ser enviada para impressão. O maior tempo, é para o armazenamento de desenhos.

Em todas as aplicações, o usuário precisa ter acesso ao desenho em um dispositivo, usando as capacidades do mesmo. Um desenho armazenado por muito tempo, deve ainda ser visível quando restaurado no dispositivo.

O projeto de CGM permite que:

- desenhos armazenados em formato CGM sejam completamente independentes de dispositivo, evitando problemas de formato de arquivo e incompatibilidade de dispositivo;

- quaisquer funções específicas de dispositivo, sejam armazenadas de uma maneira tal, que permita ao sistema de visualização ignorá-las quando seu uso seja inapropriado;
- cada arquivo CGM se auto-identifique, em relação ao gerador e a versão de CGM usada, o que, junto com a informação de "status" de CGM, fornece a melhor garantia de que mecanismos de visualização ainda existirão quando desenhos CGM sejam recuperados de arquivos ancestrais.

5.6.8.3 - TEXTO E GRÁFICOS MISTURADOS

Documentos que contêm texto e gráficos, podem ser armazenados em uma forma de "revisão", adequada para entrada em um sistema, ou em uma forma de "não revisão", como o que sai de um sistema.

Um documento de "revisão", pode conter desenhos em formato CGM, ou instruções que o permitam juntar com desenhos de outros arquivos em formato CGM. Desenhos gerados por computador ou entrados por um "scanner", são semelhantemente armazenados usando codificação de caracteres ou codificação binária; desenhos também poderiam ser preparados a mão usando codificação texto.

Um documento em forma de "não revisão", permite:

- usar apenas informação gráfica, o texto estaria em um formato diferente;

- que cada página do documento seja totalmente convertida a um mapeamento de bits (bitmap). CGM pode armazenar a sequência de páginas como desenhos;
- armazenar tanto texto como gráfico, usando os vários elementos de atributo de texto para obter diferentes tipos de letras, conjuntos de caracteres e tamanhos de caracteres.

5.7 - CGI - "COMPUTER GRAPHICS VIRTUAL DEVICE INTERFACE"

5.7.1 - INTRODUÇÃO

O projeto da Interface de Dispositivo Virtual de Computação Gráfica CGI ("Computer Graphics Virtual Device Interface"), tem como objetivo, decompor as aplicações dos usuários em funções realizáveis por hardware. A complexidade do projeto da interface varia em função da tecnologia do dispositivo.

A natureza "virtual" (ser programável ou acessível por software) da interface, tem sido mais importante do que como a interface programa e controla cada dispositivo de entrada, ou de saída, através das controladoras de dispositivo ("device driver"). As funções da interface são transparentes para o usuário.

A figura (Fig. 5.9) a seguir, mostra a posição de CGI dentro de um sistema gráfico.

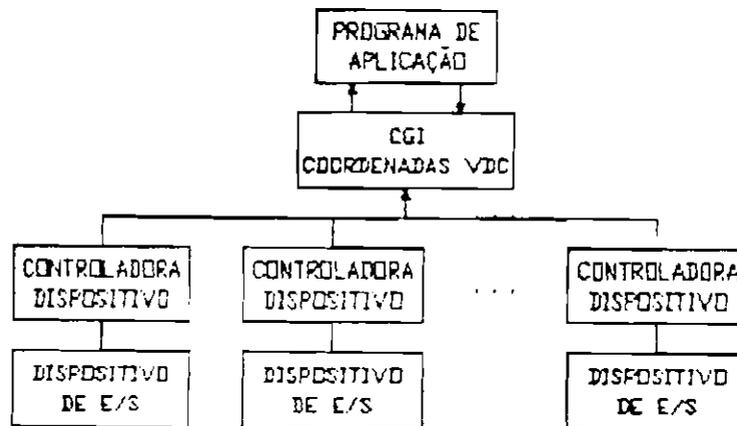


Fig. 5.9 - CGI dentro de um sistema gráfico.

O projetista de um sistema de suporte gráfico se preocupa em manter a independência do dispositivo da interface. As funções gráficas diretas e mais simples, são escritas diretamente nas controladoras de dispositivo.

Funções mais complexas são simuladas pela interface CGI, decompondo-as em várias funções simples suportadas pelas controladoras de dispositivo.

Uma das complicações, na definição das facilidades de implementação com interface, é como traduzir dependências tecnológicas na interface de usuário em funções de baixo nível, quando as tecnologias envolvidas e disponíveis não são compatíveis entre si.

Em 1977, o sistema CORE (seção 5.5) apresentou um modelo de um sistema gráfico independente de dispositivo, no qual as dependências de dispositivo podiam ser isoladas em controladoras de dispositivo, e as funções independentes de dispositivo poderiam ser compartilhadas entre várias implementações do pacote.

Porém, CORE não conseguiu identificar quais dependências de dispositivo poderiam ser isoladas em uma Interface Independente/Dependente de Dispositivo (DI/DD).

5.7.2 - HISTÓRIA DE CGI

ANSI propôs normalizar uma interface DI/DD e um metarquivo no mesmo nível. O nome Interface de Dispositivo Virtual VDI ("Virtual Device Interface"), substituiu o nome interface DI/DD para indicar que a norma descreveria um dispositivo gráfico abstrato e geral, que poderia ser realizado pela combinação de hardware, firmware e software.

Esperando ser uma interface programável, a proposta do projeto definiu VDI como uma especificação sintática e funcional padronizada do controle e intercâmbio de dados entre software gráfico independente de dispositivo e uma, ou mais, controladoras de dispositivo gráfico dependentes de dispositivo.

Foi decidido que o dispositivo virtual só aceitaria dados 2D; os dispositivos de entrada e saída seriam considerados classes separadas de dispositivos; e o conceito de pacotes de funções relacionadas ("option sets") foi adotado para suportar as diferentes implementações do dispositivo virtual.

O conceito de funções relacionadas, e não o conceito de níveis de hierarquia, foi escolhido, devido a ser difícil escolher uma hierarquia para classes de operações não relacionadas e para características de dispositivo (tais como cor, controle de rastreamento, "display" dinâmico, etc.).

Finalmente, houve o compromisso que as normas CGI e CGM seriam idênticas na especificação de elementos comuns, por terem sido baseados em um modelo de trabalho e base de desenvolvimento comuns.

Em 1985, VDI foi adotado por ISO como o WD (projeto de trabalho) inicial de CGI.

5.7.3 - ESTRUTURA DE CGI

5.7.3.1 - CONCEITOS GEOMÉTRICOS

Todas as primitivas geométricas de CGI são expressas no espaço de coordenadas de dispositivo virtual VDC, um sistema de coordenadas cartesianas 2D.

O usuário de CGI define uma janela retangular no espaço VDC. A janela especificada é mapeada em uma área retangular dentro da superfície de visualização física conhecida como vista.

A vista, em coordenadas VDC, é então transformada para o espaço de coordenadas de dispositivo DC, de tal maneira que a resolução exata do dispositivo não é mais importante.

CGI também permite o recorte da vista selecionada dentro da superfície de visualização.

5.7.3.2 - CONTROLE DE DISPOSITIVO

CGI proporciona funções para inicializar e terminar uma sessão CGI, e resetar os atributos e funções de controle a seus estados "default".

O usuário pode escolher se a implementação "bufferiza" (armazenar dados na memória) a saída eficientemente, e atualiza periodicamente o desenho, ou se força o dispositivo a atualizar o desenho continuamente.

CGI inclui funções de início e fim de desenho. Estas funções são definidas para atender os dispositivos de "hard-copy" (impressoras e plotters) e "soft-copy" (transferência de desenhos de arquivos para a superfície de visualização), os quais atuam diferentemente no início, e no fim, de uma sequência do desenho.

Os dispositivos de "soft-copy", geralmente apagam a superfície de visualização no início do desenho e não atuam no fim, deixando a imagem no vídeo.

Ao contrário, os dispositivos de "hard-copy", geralmente precisam de uma ação especial no início do desenho, se formatos de impressão devem ser carregados, e de um marcador explícito de fim de desenho, para logo iniciar a impressão.

CGI precisa de um delimitador tanto no início como no fim de um desenho, isto para liberar o usuário de se envolver com o controle a nível de dispositivo.

5.7.3.3 - PRIMITIVAS GRÁFICAS

CGI contém um conjunto de funções de desenho. Todas as primitivas gráficas e atributos de CGM tem sido incorporadas em CGI, atendendo ao desenvolvimento paralelo das normas para a interface de dispositivo virtual e os metarquivos.

Além destas, CGI implementou a função para preenchimento de áreas fechadas, devido a esta operação ser fortemente dependente de dispositivo.

5.7.3.4 - ARMAZENAMENTO DE PRIMITIVAS

CGI proporcionam-se dois tipos de armazenamento de primitivas de imagem: segmentação e raster.

Segmentação, é o mecanismo para o armazenamento e manipulação de grupos de primitivas gráficas, parametrizadas ou não, que constituem um todo (segmento).

Raster, é o mecanismo para o armazenamento ou manipulação de imagens (formadas por primitivas gráficas), ou partes de imagens, previamente realizadas.

O modelo de segmento de CGI é similar porém mais simples que o modelo de segmento de GKS. Os segmentos podem ser transformados, tornados invisíveis, realçados, ordenados, e ser detetáveis ou não. Estas propriedades estão associadas a todo o segmento e apenas alteram a interpretação do conteúdo do segmento não afetando sua descrição de armazenamento.

CGI não permite a edição seletiva de elementos do segmento. Um segmento apenas pode ser renomeado, apagado, reaberto para anexar elementos, ou ser copiado dentro de outro segmento. Nenhuma descrição hierárquica é mantida, isto é, nenhum elemento que indique que um segmento referencia outro é armazenado.

Operações raster permitem armazenar e modificar imagens gráficas. Representam o segundo nível de armazenamento. Suas unidades básicas são: o "pixel" (elemento de imagem) e o "bitmap" (mapeamento de bits), o qual é uma arranjo retangular de "pixels".

"Bitmaps" definem partes de uma imagem em forma raster e são criados definindo uma região particular do espaço VDC, de acordo com o mapeamento atual do espaço VDC com as coordenadas físicas do dispositivo DC. Este mapeamento, define a posição em coordenadas DC e o número de pixels em x e y necessários para conter a região solicitada.

Qualquer mudança subsequente no mapeamento VDC para DC não afetará no número de pixels envolvidos nos bitmaps existentes, embora isto afete a área que os pixels representam em VDC.

Uma vez criado um "bitmap", este pode ser selecionado e exibido como saída gráfica. A cada "bitmap" pode ser associado um nome. Assim, estes bitmaps podem ser combinados, usando operações lógicas, e mostrados na superfície de exibição.

5.7.3.5 - FUNÇÕES DE ENTRADA DE CGI

As funções de entrada de CGI (dispositivos lógicos de entrada estão projetadas para suportar os modelos de entrada de outros padrões, e dividem-se em várias classes de entrada.

As classes de entrada suportadas são as seis padronizadas em GKS (seção 5.8) e propostas para GKS-3D e PHIGS: seletor ("choice"), posicionador ("locator"), apontador lógico ("logical pick"), cadeia de caracteres ("text"), cadeia de posições ("stroke") e quantificador ("valuator").

Dispositivos destas classes podem ser suportadas em 4 modos: demanda ("request"), amostragem ("sample"), evento ("event"), e demanda de eco ("echo request"). Os 3 primeiros modos estão incluídos no GKS.

Todas as funções de entrada de CGI operam em modo síncrono. Qualquer operação assíncrona é suportada através de processos locais independentes respondendo sincronamente aos comandos CGI.

Cada dispositivo lógico inicializado, tem um processo de monitoração associado, cuja função é manter um valor atualizado do dispositivo físico associado.

5.7.3.6 - LIGAÇÕES ("BINDINGS")

CGI suporta ligações ("bindings") com sistema de suporte gráfico padrões tais como GKS, GKS-3D e PHIGS, através de chamadas a funções de biblioteca. CGI pode ser linkado juntamente com cada um destes padrões.

Em contraste, todas as atuais ligações com CGM, não são através de chamadas de funções, devido a que a intenção de CGM é armazenar e transferir descrições de desenhos e não invocar mecanismos de execução.

Para facilitar seu uso em sistemas distribuídos, os dados transmitidos por CGI serão modelados em CGM permitindo a migração entre estas duas normas.

5.7.3.7 - TRATAMENTO DE ERROS

A filosofia de erros de CGI, é minimizar o diálogo entre o dispositivo e o usuário quando o erro ocorre.

Ao contrário de GKS, o qual força à programação correta especificando um grande número de condições de erros detetáveis e as específicas reações a estes, CGI apenas proporciona diretrizes para uma programação recomendável que não conduza a erros muito sérios.

Quando CGI detecta certos erros, a interface escolhe uma ação de correção e procede sem notificar ao usuário. Por exemplo, poderia substituir um parâmetro de atributo não suportado, ou fora de um intervalo válido, por um valor de parâmetro de atributo válido.

A diferença deste tratamento de erros em relação ao GKS, tem como base as diferenças que existem entre a comunidade de usuários de GKS e CGI.

GKS é uma interface de alto nível para programar aplicações gráficas, CGI é uma interface interna. Um ambiente CGI, uma vez estabelecido é mais estático. Os usuários de CGI são programadores de aplicações os quais estão mais aptos para gerenciar seus próprios erros de processamento conforme a necessidade.

5.8 - GKS - "GRAPHICAL KERNEL SYSTEM"

5.8.1 - INTRODUÇÃO

Apresenta-se o GKS, porém somente sua semântica é descrita. A sintaxe das funções foge do escopo desta abordagem.

5.8.1.1 - DEFINIÇÃO DE PACOTE GRÁFICO

Denomina-se pacote gráfico ao conjunto de funções, as quais, dentro de um sistema de programação permitem o controle dos dispositivos gráficos de entrada e saída de um sistema gráfico.

Estas funções são manipulados por software e estão incorporadas ao sistema operacional do equipamento. Através do software do pacote, se realiza a comunicação com os diversos periféricos gráficos.

5.8.1.2 - REQUISITOS DE UM PACOTE GRÁFICO

A elaboração de um pacote gráfico é difícil e complexa em função da diversidade de aplicações e equipamentos. A evolução das aplicações e equipamentos obrigam que esta interface de software seja revisada, estendida e reformulada. Apesar disto, os princípios gerais como requisitos de um pacote gráfico são:

1. Consistência

Não pode existir contradição entre as diversas unidades e conceitos do pacote gráfico.

2. Completeza

O pacote gráfico deve incluir recursos suficientes para atender à maioria das aplicações e suportar a maioria dos dispositivos gráficos atuais.

3. Simplicidade

Funções e conceitos confusos devem ser excluídos.

4. Clareza

Os conceitos e modelos do pacote, devem estar expressados claramente e as diversas interdependências entre as funções devem estar bem definidas.

5. Robustez

O pacote deve proteger o programa de aplicação, que usa o pacote, em relação a erros provocados por mal uso das funções, ou por falhas do sistema ou do equipamento.

6. Independência de dispositivos

As funções do pacote gráfico devem ser elaboradas de tal maneira que os programas de aplicação tenham acesso a recursos de diferentes dispositivos gráficos sem a necessidade de mudanças estruturais no pacote.

7. Implementabilidade

A implementação do pacote deve ser compatível com o sistema operacional, tal que as linguagens de programação que rodam sob o sistema operacional, possam ter acesso as funções do pacote. Assim mesmo deve suportar a maioria dos dispositivos gráficos.

8. Eficiência

A implementação e instalação do pacote, deve garantir um desempenho satisfatório das suas diversas funções sem a necessidade de um ambiente especial.

5.8.2 - O GKS

O Núcleo de Sistema Gráfico GKS ("Graphical Kernel System"), é um pacote gráfico aprovado pela ISO em 1984 e por outras diversas entidades nacionais de padronização como a DIN, ANSI, BSI, etc.

Além de possuir os requisitos básicos de um pacote, como especificados acima (seção 5.8.1), o sistema tem os seguintes objetivos:

- Geração e representação de desenhos.
- Transformação dos desenhos criados no sistema de coordenadas da aplicação WCS aos diferentes dispositivos de exibição gráfica, mapeando os desenhos na superfície de exibição destes dispositivos.
- Controle funcional dos dispositivos de entrada e saída gráfica conectados ao sistema.
- Estruturação de desenhos em partes manipuláveis independentemente.
- Armazenamento e transmissão de desenhos.

Um programa em uma determinada linguagem de programação, usando funções próprias da linguagem, pode acessar o núcleo. Para este propósito, GKS propõe padrões para estas ligações ("bindings") com as principais linguagens de programação, como por exemplo, FORTRAN, Pascal, C e Modula II.

A primeira versão do GKS é um pacote 2D. Devido a um sistema 2D ser suficiente para a maioria das aplicações, não foi incluída no GKS a manipulação de elementos geométricos 3D para não comprometer o desempenho da implementação. Não entanto, está em andamento a especificação do GKS 3D, uma extensão 3D do GKS.

Como todo pacote gráfico, o GKS se posiciona entre o programa de aplicação e os diversos periféricos gráficos do sistema.

A especificação do GKS permite tornar os programas de aplicação independentes das características específicas e dos códigos de controle dos periféricos de uma particular instalação.

Assim, para o programa de aplicação ficam definidos periféricos virtuais que se comportam de uma maneira uniforme. Logo, o acesso e controle de um "plotter" é idêntico ao de um terminal de vídeo.

Entretanto, quando o programa de aplicação precisar de informações específicas dos periféricos (como saber se um dispositivo de saída é matricial ou vetorial), o programa pode usar "funções de consulta", através das quais, o GKS fornece os valores dos parâmetros de instalação e do estado do sistema.

5.8.3 - ESTRUTURA DO GKS

A estrutura do GKS com as características de independência de periféricos, decorre da introdução dos seguintes conceitos:

1. Saída gráfica

Os desenhos criados através do GKS, são formados por elementos básicos chamados "primitivas gráficas de saída". Uma série de atributos (cor, textura, etc.), cujos valores estão sob o controle do programa de aplicação, controlam o aspecto visual de exibição das primitivas.

2. Sistemas de coordenadas

As entidades geométricas são expressas pelo programa de aplicação em um, ou mais, sistemas de coordenadas. O GKS fornece funções de transformação para mapear as entidades no sistema de coordenadas de cada dispositivo de exibição.

3. Estação gráfica de trabalho

GKS permite, a nível de implementação, a integração de dispositivos de entrada e saída gráfica em uma mesma unidade, configurando uma "estação gráfica de trabalho". Assim mesmo, podem ser generalizados diversos recursos a todas as estações de trabalho, independente dos dispositivos físicos utilizados para tal fim.

4. Segmentação

Os desenhos criados pelo programa de aplicação podem ser agrupados em unidades chamadas "segmentos". Através dos segmentos, os desenhos podem ser manipulados independentemente (exibidos, apagados, geometricamente transformados, selecionados, etc.).

Embora a segmentação seja um recurso próprio das estações de trabalho, existe um recurso adicional determinado pela introdução do Armazenador de Segmentos Independente de Estação de Trabalho WISS ("workstation-independent segment storage"), que permite que os segmentos sejam copiados e transferidos para outras estações de trabalho.

5. Entradas gráficas

Além dos dispositivos de entrada de dados geométricos (como par de coordenadas), o GKS suporta outros dispositivos lógicos de entrada que incluem entradas alfanuméricas, de seleção (botões) e de valores reais (potenciômetro e alavancas). Através destas entradas, o implementador da estação de trabalho mantém uma comunicação com o programa de aplicação.

6. Metarquivo ("metafile")

Os desenhos armazenados em segmentos, só existem até que o programa que os criou os destrua ou finalize. Os metarquivos, permitem armazenar os segmentos em arquivos e a transmissão de desenhos e imagens geradas durante a execução.

Os desenhos ou imagens guardados em metarquivos podem ser depois recuperados, permitindo salvar uma sessão interativa.

7. Tabelas descritivas e listas de estado

O GKS controla a execução do programa de aplicação através dos valores de uma série de variáveis internas organizadas nas chamadas "listas de estado", as quais contém informações tais como quais estações de trabalho se encontram ativas em um determinado ponto da execução.

As "tabelas descritivas" contém informações estáticas da configuração do sistema, tais como os recursos de entrada disponíveis em cada uma das estações de trabalho. O programa de aplicação tem acesso a esta estrutura de dados através das funções de consulta ("inquire functions") do GKS.

8. Níveis de implementação

Em função das necessidades de certas áreas de aplicação e as facilidades existentes nas várias instalações, o GKS pode ser implementado em 9 níveis que representam subconjuntos de toda a complexidade do sistema.

O nível mínimo, fornece apenas recursos de saída gráfica sem estrutura de segmentos e adequa-se a aplicações modestas e não interativas. O GKS da ANSI especifica mais 3 níveis de implementação abaixo dos 9 níveis do GKS da ISO.

9. Tratamento de erros

Diante de situações de erro, como por exemplo, tentar transladar um segmento inexistente, o GKS reage de uma forma padronizada gerando um arquivo de mensagens de erros. Este tratamento de erros padronizado pode ser alterado pelo programa de aplicação.

5.8.4 - INICIALIZAÇÃO DO GKS

A função OpenGKS inicializa o GKS e torna suas funções disponíveis. Esta função recebe como parâmetros o identificador de um arquivo, onde o GKS gravará as mensagens de erro, e a quantidade de memória que poderá ser alocada como área interna de trabalho.

Esta função inicializa as tabelas descritivas e as listas de estado. Alguns dos valores iniciais de variáveis destas estruturas de dados são dependentes da instalação.

Após encerrar os recursos gráficos do GKS, a função CloseGKS encerra a sessão fechando o arquivo de mensagens de erros e liberando a memória alocada por GKS. Após isto, as listas de estado e as tabelas descritivas deixam de existir por serem dinâmicas.

5.8.5 - PRIMITIVAS GRÁFICAS DE SAÍDA DO GKS

Programas de sistemas gráficos geralmente têm desenhos ou imagens como saída, e podem utilizar um ou mais dispositivos de entrada de dados.

As imagens produzidas pelo programa e apresentadas nas superfícies de exibição dos dispositivos gráficos de saída, são formadas por retas nos sistemas vetoriais e por "pixels" nos sistemas matriciais. Estes elementos básicos são as primitivas gráficas do dispositivo, podendo incluir caracteres e símbolos como componentes de uma imagem.

O GKS dispõe de 6 primitivas gráficas de saída: traçado de linhas ("polyline"), marcador de pontos ("polymarker"), texto ("text"), duas orientadas para dispositivos matriciais, áreas poligonais e retângulos ("cell array" e "fill area"), e uma primitiva de propósito geral GPD ("generalized drawing primitive").

As primitivas gráficas de saída do pacote, devem poder ser representadas em qualquer dispositivo de saída gráfica disponível no sistema, independente de suas características.

5.8.5.1 - "POLYLINE"

Esta primitiva gera uma polilinha unindo uma dada sequência de pontos no plano com segmentos de reta.

A função Polyline tem como parâmetros o número de pontos a serem unidos e um vetor 2D contendo as coordenadas destes pontos. Assim mesmo, representa a primitiva básica do GKS para o traçado de retas.

O traçado de curvas, tais como arcos e circunferências, pode ser obtido através de uma série de pequenos segmentos de reta. Dependendo da implementação, estas curvas também podem ser geradas usando a primitiva GPD.

Os atributos da primitiva "polyline" são:

1. Tipo de linha

Especifica a textura do traçado de cada segmento de reta. Este atributo é setado pela função SetLineType e são 4 os tipos de linha predefinidos: contínuo, tracejado, pontilhado e traço-ponto.

2. Fator de escala da espessura

Controla a espessura do traçado de cada segmento. Este atributo é setado pela função SetLineWidthScaleFactor.

3. Índice de cor

Controla a cor dos segmentos de reta da "polyline". Este atributo é setado pela função SetPolylineColourIndex.

4. Índice

Associa um índice a "polyline" através do qual esta primitiva pode ser referenciada.

Embora o GKS permita setar estes atributos em cada dispositivo de saída, estes atributos podem ser associados globalmente à primitiva, isto é, a primitiva é desenhada com os valores correntes de seus atributos.

Os atributos das primitivas de saída podem ser de 2 tipos: atributos geométricos e atributos não geométricos.

Os atributos geométricos, estão sujeitos à todas as transformações geométricas que se aplicam sobre as figuras. Rotações e mudanças de escala, por exemplo, afetam os pontos que determinam a "polyline". Por tanto, os pontos que compõem uma "polyline" são atributos geométricos.

Os atributos não geométricos, são aqueles imunes às transformações. Como exemplo temos o índice de cor, a espessura e a textura da "polyline". Assim a ampliação de uma figura não resultará no aumento da espessura da "polyline", por exemplo.

A associação de atributos as primitivas pode ser: modal, estática ou direta. É modal quando se usa os atributos correntes, é estática quando não ha mais a possibilidade de alterar os atributos a posteriori, e é direta quando os valores dos atributos são fornecidos diretamente no momento da chamada da função de atribuição.

5.8.5.2 - "POLYMARKER"

Esta primitiva permite marcar pontos na superfície de exibição dos dispositivos de saída gráfica. As marcas são figuras centradas em cada ponto de um vetor passado à função Polymarker como argumento. Exemplos de marcas são o ponto, a cruz, o asterisco, o círculo e o x.

Portanto, os argumentos da função Polymarker são o número de pontos onde colocar as marcas e um vetor 2D contendo as coordenadas destes pontos.

Os atributos da primitiva "polymarker" são:

1. Tipo da marca

Determina a figura utilizada como marca. Os valores pré-definidos são: 1=ponto (.), 2=cruz (+), 3=asterisco (*), 4=círculo (O) e 5=x (X). Este atributo é setado pela função SetMarkerType.

2. Fator de escala de tamanho

Controla o tamanho da marca. Este atributo é setado pela função SetMarkerSizeScaleFactor.

3. Índice de cor

Controla a cor da marca. Este atributo é setado pela função SetPolymarkerColourIndex.

5.8.5.3 - "TEXT"

Informações textuais frequentemente acompanham as informações gráficas, por exemplo, os títulos em mapas. O GKS provê a função Text para este propósito.

A função Text tem como parâmetros o ponto de inserção do texto e a cadeia dos caracteres a ser traçada.

A primitiva "text" possui mais nove atributos que determinam as características do texto inserido:

1. Precisão

Determina a qualidade com que os atributos geométricos do texto são desenhados.

2. Fonte

Seleciona o fonte dos caracteres do texto, tais como os fontes ASCII, românico, itálico, cursivo, grego, etc. Fonte e precisão formam um único atributo e são setados pela função SetTextFountandPrecision.

3. Altura

Determina a altura dos caracteres. E setado pela função SetCharacterHeight.

4. Vetor de orientação

Controla a direção vertical de cada caracter do texto. Através deste vetor o texto pode ser inclinado. A função que seta este atributo é `SetCharacterUpVector`.

5. Direção

Controla a direção com que cada caracter é posicionado em relação aos precedentes. Pode assumir os valores: para direita, para esquerda, para cima e para baixo. A função que seta este atributo é `SetTextPath`.

6. Alinhamento

Posiciona o ponto de inserção do texto dentro do texto. Na horizontal pode ser centralizado, à esquerda ou à direita. Na vertical pode ser superior, inferior e meio. A função que seta este atributo é `SetTextAligment`.

7. Fator de expansão

Define a relação largura/altura do quadrado que encaixa cada caracter. Este atributo é setado pela função `SetCharacterExpansionFactor`.

8. Espaçamento

Controla a distância entre dois caracteres. Valores negativos determinam uma superposição parcial ou total. A função `SetCharacterSpacing` controla este atributo.

9. Índice de cor

Controla a cor dos caracteres. A função `SetTextColourIndex` seta este atributo.

5.8.5.4 - TRATAMENTO DE CORES

Cor é um atributo presente em todas as primitivas gráficas de saída e dependente do dispositivos de saída em questão, pois alguns dispositivos de saída são monocromáticos e outros coloridos, capazes de representar desde dezenas até milhões de diferentes tons de cores.

GKS soluciona esta questão associando às primitivas de saída um "índice de cor" e criando em cada terminal gráfico de saída uma "tabela de cores", de tal maneira, que quando uma primitiva gráfica de saída é direcionada a um terminal gráfico, o índice de cor da primitiva irá buscar na tabela de cores do terminal a cor com a qual a primitiva será traçada.

A figura (Fig. 5.10) a seguir, representa este processo:

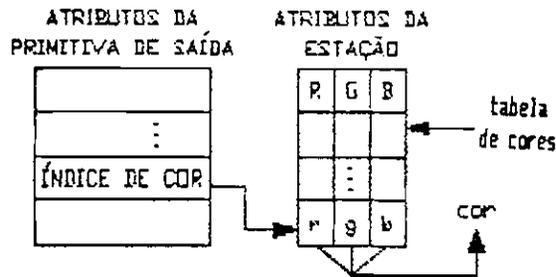


Fig. 5.10 - Indexação da tabela de cores.

Dependendo do conteúdo das diferentes tabelas de cores dos terminais gráficos (dispositivos de saída), a mesma primitiva pode ter a cor verde em um terminal, um tom de cinza em outro, e em um terceiro a única cor disponível.

Assim, o programa de aplicação se ajusta as capacidades de representação de cores dos dispositivos de exibição. Os conteúdos das tabelas de cores podem ser mudados pelo programa de aplicação, porém a tabela de cores é única em cada terminal.

A representação de cores no GKS usa o modelo de composição de cores primárias. Cada tom é resultado da mistura de frações (entre 0 e 1) das cores básicas puras: vermelho, verde e azul. Logo, cada tom é representada pela tripla (r,g,b) . Como exemplo, a tripla $(0,0,0)$ representa o preto e $(1,1,1)$ o branco.

A função `SetColourRepresentation` permite alterar a tabela de cores. Esta função tem como argumentos o identificador da estação de trabalho, que terá sua tabela de cores mudada, a posição da cor na tabela de cores, e três números reais que representam as frações da tripla (r,g,b) .

Alguns terminais gráficos de vídeo já incorporam em hardware a tabela de cores. Naqueles em que não está disponível, o GKS simula internamente a tabela de tal modo que para o usuário este mecanismo pareça presente em todos os dispositivos de saída.

5.8.5.5 - "CELL ARRAY"

As três primitivas acima descritas, podem ser representadas tanto em dispositivos vetoriais quanto em matriciais.

Para suportar a crescente difusão dos dispositivos matriciais que permitem a representação de regiões, o GKS justificou a inclusão de duas primitivas: "cell array" e "fill area", que atendem especialmente aos dispositivos matriciais.

A primitiva "cell array" permite descrever imagens matriciais no GKS. A imagem matricial é representada por uma matriz retangular de células às quais pode ser atribuído independentemente um índice de cor. Observe a figura (Fig. 5.11) abaixo:

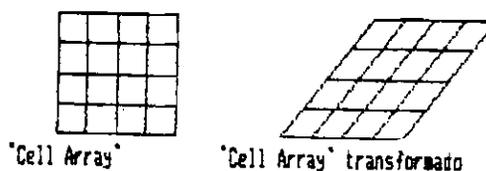


Fig. 5.11 - Primitiva "cell array".

Para a representação desta região a função `CellArray` tem como argumentos as coordenadas dos vértices opostos do retângulo, os incrementos `dx` e `dy` da matriz de células e um vetor 2D que associa um índice de cor a cada célula.

Diminuindo `dx` e `dy` consegue-se que o centro de cada célula coincida com um "pixel" do dispositivo matricial de saída. Desta forma pode ser construída uma imagem com controle total sobre a cor de cada "pixel".

Como a área retangular do "cell array" é um atributo geométrico, a primitiva fica sujeita às transformações geométricas que afetam as outras primitivas. Portanto, pode haver um desajuste entre células e "pixels" como mostra a figura (Fig. 5.11) anterior.

5.8.5.6 - "FILL AREA"

Esta primitiva é definida para tratar áreas de imagens. As áreas são delimitadas por uma poligonal fechada e podem ser preenchidas.

A figura (Fig. 5.12) a seguir, ilustra o conceito usado para o preenchimento de áreas fechadas por uma poligonal.

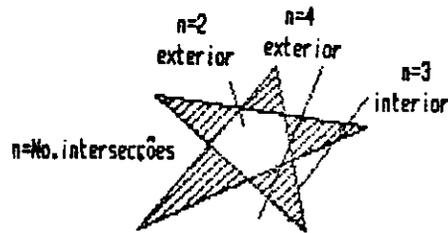


Fig. 5.12 - Preenchimento de areas por "fill area" usando o método da paridade.

Um ponto é interior a área (e portanto preenchido), se e somente se, uma semireta emanando deste ponto cruza os lados da poligonal um número ímpar de vezes.

A função FillArea permite o preenchimento de áreas, e tem como argumentos o número de vértices do polígono e um vetor 2D contendo as coordenadas dos vértices. Após unir os vértices com retas, a função aplica o algoritmo de preenchimento na poligonal fechada resultante.

O tipo de preenchimento das áreas, é controlado pelo parâmetro "estilo de preenchimento de área" o qual pode ser:

1. Oco ("hollow")

Só é traçada a poligonal com o índice de cor de preenchimento. A área interior não é preenchida.

2. Sólido ("solid")

O interior é preenchido uniformemente com o índice de cor de preenchimento.

3. Hachurado ("hatch")

O interior é hachurado com o índice de cor de preenchimento. O tipo de hachura é dado pelo índice de estilo de preenchimento.

4. Estampado ("pattern")

O interior da área é preenchido regularmente com um padrão retangular que corresponde a uma entrada na "tabela de padrões" apontada pelo índice de estilo de preenchimento. Cada padrão é formado por um arranjo matricial de índices de cores à semelhança da "cell array".

Cada dispositivo de saída pode ter uma tabela de padrões diferente dos outros dispositivos, assim uma mesma área pode ser representada com padrões diferentes em cada dispositivo ativo.

A primitiva "fill area" está voltada para os dispositivos matriciais, nos quais o estilo estampado é o que mais se destaca. Os dispositivos vetoriais atendem ao estilo oco e, sob certas restrições, ao estilo hachurado.

5.8.5.7 - GPD

A primitiva generalizada de desenho GPD ("generalized drawing primitive") é um recurso padronizado para acessar primitivas especiais específicas a certos dispositivos e não previstas entre os demais.

Esta primitiva é usada para o traçado de cônicas, curvas spline e pintura de regiões. Porém, esta primitiva é dependente do hardware do dispositivo, e a portabilidade de programas usando esta primitiva fica limitada as instalações que a suportam.

A função GPD do GKS admite 4 parâmetros: o número de pontos de um vetor, um vetor 2D fornecendo as coordenadas destes pontos que serão interligados, um identificador da primitiva solicitada, e uma cadeia de caracteres contendo informações diversas necessárias ao traçado da primitiva.

5.8.6 - SISTEMAS DE COORDENADAS

5.8.6.1 - INTRODUÇÃO

Para representar as primitivas gráficas de saída é necessário fixar um sistema de coordenadas. Diferentes aplicações podem usar diferentes sistemas de coordenadas, por exemplo os sistemas de coordenadas cartesianas e polares.

Como a maioria das aplicações usa o sistema de coordenadas cartesianas, o GKS escolheu este sistema para descrever os atributos geométricos das primitivas de saída e o denominou de sistema de coordenadas do mundo WCS ("world coordinate system"). O sistema de coordenadas preferido pela aplicação, que é transformado para o sistema WCS, é chamado de sistema de coordenadas do usuário UCS ("user coordinate system").

5.8.6.2 - COORDENADAS NORMALIZADAS

Os dispositivos de exibição, usados para representar as primitivas gráficas, usam diferentes sistemas de coordenadas (em geral cartesiano), posicionam a origem em diferentes posições dentro da tela, o eixo vertical é definido para cima ou em sentido inverso, enfim, são totalmente incompatíveis entre si.

Para ter uma independência destes fatores, o GKS define uma "superfície de exibição virtual" representada por um quadrado unitário com vértices em (0,0) e (1,1).

As coordenadas neste quadrado são denominadas coordenadas de dispositivo normalizadas NDC ("Normalized Device Coordinates"). O quadrado é uma "tela virtual" onde as figuras formadas por primitivas gráficas compõem uma "imagem virtual".

Para representar as primitivas gráficas de saída, expressas em coordenadas WCS, nesta tela virtual, é estabelecida uma transformação linear entre os dois espaços.

Esta transformação define um retângulo ou janela de normalização ("window") em coordenadas WCS, e o mapeia em um segundo retângulo ou vista de normalização ("viewport") na tela virtual, como mostra a figura (Fig. 5.13) a seguir:

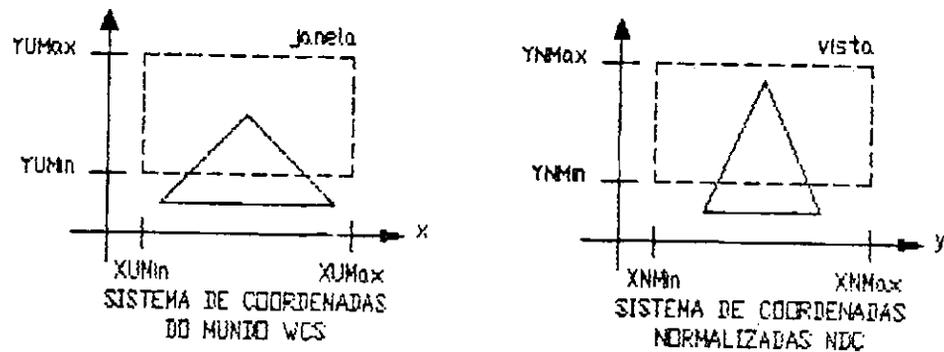


Fig. 5.13 - A transformação de normalização.

Este mapeamento linear entre os dois espaços transforma um ponto (x_U, y_U) em coordenada do mundo para um ponto (x_N, y_N) na tela virtual, conforme as equações:

$$x_N = x_{NMin} + ((x_U - x_{UMin}) / (x_{UMax} - x_{UMin})) * (x_{NMax} - x_{NMin}) \quad (5.1)$$

$$y_N = y_{NMin} + ((y_U - y_{UMin}) / (y_{UMax} - y_{UMin})) * (y_{NMax} - y_{NMin})$$

Esta transformação que define todo o espaço WCS e não apenas a janela, é denominada "transformação de normalização". Envolve apenas uma translação e escala e não inclui rotações. Se a janela e a vista apresentam fatores de forma (relação altura/largura) distintos, a transformação de escala em cada um dos eixos é distinta.

Uma vez definida a transformação de normalização através da janela e da vista, esta passa a ser a transformação corrente e todas as primitivas gráficas de saída do espaço WCS são mapeadas para o espaço NDC.

Esta transformação pode ser redefinida, permitindo uma flexibilidade aos programas de aplicação, e não afetando as primitivas já desenhadas no espaço NDC.

O GKS armazena uma lista de transformações de normalização que são ativadas, onde o número máximo de elementos desta lista depende do nível da implementação.

As funções SetWindow e SetViewport, permitem definir a janela de normalização em coordenadas WCS e a vista de normalização em coordenadas NDC, respectivamente. A função SelectNormalizationTransformation permite selecionar a transformação de normalização corrente.

O GKS permite que as primitivas que seriam exibidas fora, ou parcialmente fora, da vista sejam eliminadas total, ou parcialmente, da vista. Este procedimento é conhecido como recorte ("clipping"). A função SetClippingIndicator permite que a exibição das primitivas de saída estejam sujeitas a recorte ou não.

Internamente, o GKS preserva a integridade de toda a primitiva gráfica de saída portanto estas não são afetadas pelo recorte.

5.8.7 - SEGMENTAÇÃO

A construção de figuras através das primitivas do GKS as quais aplica-se uma transformação de normalização para logo ser posicionadas na superfície de exibição, implica em que as figuras não podem ser modificadas, eliminadas ou reposicionadas a menos que toda a imagem seja apagada e reconstruída.

Esta restrição torna o sistema gráfico insatisfatório para aplicações interativas onde o usuário da estação gráfica deseje alterar ou eliminar partes selecionadas do desenho.

Para atender a este problema, o GKS criou a definição de "segmentos". O segmento consiste da agrupação de um conjunto de primitivas de saída formando uma unidade logicamente definida que possui um nome identificável.

O GKS também associa atributos aos segmentos. Estes atributos permitem que o programa de aplicação controle a visibilidade, localização, realce, prioridade e detectabilidade das primitivas contidas no segmento.

5.8.7.1 - CONSTRUÇÃO DE SEGMENTOS

Após um segmento ser criado pela função CreateSegment, este passa a ser o segmento aberto e todas as primitivas gráficas de saída além de serem exibidas são incluídas no segmento até este ser fechado através da função CloseSegment.

Após fechar o segmento, as primitivas subsequentes não são mais incluídas naquele segmento e não há mais segmento aberto. Enquanto um segmento está aberto não é possível criar outro segmento. Uma vez fechado o segmento, este não pode ser reaberto e nenhuma primitiva pode ser nele incluída ou eliminada.

A função DeleteSegment elimina um segmento e todas as primitivas que o compõem. Os segmentos podem ser renomeados através da função RenameSegment.

5.8.7.2 - ATRIBUTOS DE SEGMENTOS

1. Transformação de segmentos

As primitivas de saída dentro de um segmento são mapeadas ao espaço NDC pela transformação de normalização corrente no momento de sua criação.

Além desta transformação, as primitivas dos segmentos estão sujeitas a outra transformação linear, única para todo o segmento, denominada "transformação de segmento".

A transformação de segmento é um mapeamento linear no mesmo espaço NDC, que permite, além da translação e escala, a rotação e reflexão das primitivas.

A figura (Fig. 5.14) a seguir, ilustra a sequência de transformações a que estão sujeitas as primitivas contidas nos segmentos antes de serem exibidas nos dispositivos de exibição.

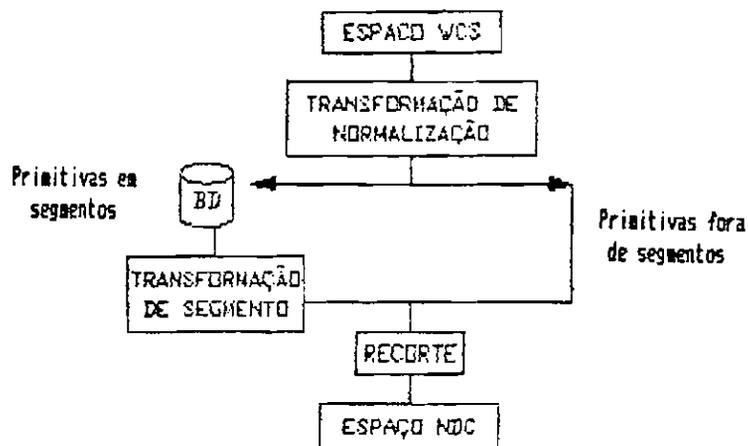


Fig.5.14 - Sequência de transformações nos segmentos antes da visualização.

2. Visibilidade

Este atributo define se as primitivas do segmento são exibidas ou não. A função `SetVisibility` controla este atributo.

3. Realce

Este atributo permite que um segmento exibido seja realçado ("highlighted") visualmente, destacando-o dos outros segmentos não realçados. O realce pode ser um aumento de intensidade na cor, uma cor especial ou um efeito de cintilação. Este atributo é controlado pela função `SetHighlighting`.

4. Prioridade de segmentos

Quando uma imagem apresenta várias regiões preenchidas criadas com as primitivas "fill area" e "cell array" e estas se sobrepõem parcial ou totalmente, o problema é determinar qual primitiva será exibida na área de interseção.

Para solucionar este impasse, o GKS define o conceito de "prioridade de segmentos", portanto o segmento que tenha maior prioridade será o que prevalecerá. A função SetSegmentPriority, permite definir a prioridade de um segmento.

5. Detetabilidade

Este atributo só faz sentido nos dispositivos lógicos de entrada do tipo apontador ("pick device"). Permite controlar se um segmento, ou parte dele, pode ou não ser selecionado pelo operador do apontador. A função SetDetectability, permite controlar este atributo.

5.8.7 - DISPOSITIVOS LÓGICOS DE ENTRADA

Os dispositivos gráficos físicos de entrada são associados a "dispositivos lógicos de entrada" DLE, possibilitando o uso destes dispositivos de uma forma independente de configuração.

Assim, um programa de aplicação que precisa dispor de um determinado tipo de entrada pode fazer referência a um DLE sem ter que se preocupar com o meio físico (dispositivo) por intermédio do qual a entrada é adquirida.

Na verdade, como exemplo, conforme mostra a figura (Fig. 5.15) abaixo, são definidas duas janelas em coordenadas WCS, uma de vídeo e outra de mesa; e duas vistas em coordenadas NDC, uma de vídeo e outra de mesa.

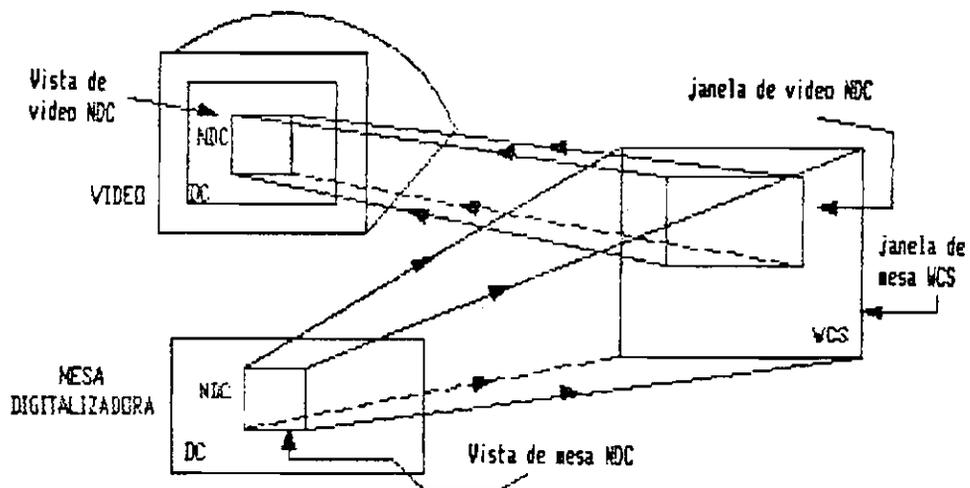


Fig. 5.15 - Transformação de coordenadas NDC para coordenadas WCS.

A janela de vídeo, em coordenadas WCS, é mapeada para a vista de vídeo em coordenadas NDC. A vista é depois convertida para coordenadas de dispositivo DC.

Por outro lado, um dispositivo lógico de entrada DLE ao qual está associado um dispositivo de entrada específico, faz o mapeamento de coordenadas de dispositivo DC do dispositivo de entrada para a vista de mesa em coordenadas NDC a qual depois é mapeada para a janela de mesa em coordenadas WCS.

No GKS os dados de entrada para um programa de aplicação são classificados em 6 tipos. Para cada tipo existe uma classe de DLE. As classes de DLE são:

1. Localizador ("locator")

Um DLE desta classe retorna para o programa de aplicação a posição de um ponto (x,y) do plano em coordenadas do mundo. A função RequestLocator, permite que o programa de aplicação solicite dados a este DLE.

Exemplos de dispositivos físicos localizadores são o "mouse", manche, "trackball" e os "thumbwheels" em um terminal de rastreamento programado.

2. Cadeia de posições ("stroke")

Um DLE desta classe, retorna ao programa de aplicação uma sequência de posições (x,y) do plano de uma vez só, evitando fazer isto através de chamadas reiteradas a um dispositivo localizador. A função RequestStroke, permite solicitar dados a este DLE.

São exemplos desta classe, os dispositivos localizadores mencionados acima mais a mesa digitalizadora.

3. Identificador ("pick")

Um DLE da classe apontador, retorna para o programa de aplicação o nome de um segmento contendo uma primitiva de saída apontada pelo dispositivo físico. A função RequestPick do GKS, permite controlar este DLE.

Como exemplos de dispositivos físicos chamados apontadores podem ser citados a caneta de luz ("light pen") e os dispositivos físicos localizados acima citados.

4. Avaliador ("valuator")

Um DLE quantificador retorna ao programa de aplicação um valor real. A função RequestValuator, permite que o programa de aplicação solicite dados a este DLE.

Quantificadores são associados a teclados e potenciômetros, ou a simulações destes dispositivos físicos.

5. Seleccionador ("choice")

Um DLE desta classe retorna para o programa de aplicação um valor pertencente a uma dada faixa de valores. A função RequestChoice, permite solicitar dados a este DLE.

Como exemplo, podem ser citados as chaves seletoras, as teclas de funções ou um dispositivo físico localizador usado para selecionar uma opção de um menu de tela.

6. Cadeia de caracteres ("string")

Um DLE desta classe retorna para o programa de aplicação uma cadeia de caracteres, através da função RequestString. Como exemplo deste tipo pode-se citar o teclado.

5.8.8.1 - MODOS DE OPERAÇÃO DOS DISPOSITIVOS LÓGICOS DE ENTRADA

Na comunicação entre o sistema gráfico e o operador na entrada de dados por meio dos dispositivos de entrada do GKS, destacam-se os seguintes elementos:

1. Sinal de atenção

O sistema manifesta a espera de entrada de dados através de um sinal de atenção, por exemplo um cursor piscante.

2. Eco

Quando o operador do sistema edita ou seleciona um valor de entrada, o sistema edita o valor corrente da entrada, isto é, a sequência editada vai sendo mostrada (ecoada) na tela.

3. Disparo

O operador aciona um dispositivo indicando o fim da seleção do valor de entrada, por exemplo aperta a tecla "return".

4. Reconhecimento

O sistema indica ao operador que recebeu a entrada de dados solicitada, por exemplo o cursor desaparece da tela.

A interação de um DLE pode se dar em 3 modos: por demanda, por amostragem ou por evento:

1. Por demanda

Neste modo, a interação de um DLE é iniciada pelo programa de aplicação através da invocação de uma função Request do GKS e termina quando o operador aciona o primeiro disparo.

Durante a interação, a execução do programa é suspensa e não podem operar dois DLE simultaneamente.

2. Por amostragem

Neste modo, a interação do programa de aplicação com o DLE é iniciada e terminada através de uma função do tipo SetDLEMode.

A leitura do valor corrente do DLE é feita através de uma função do tipo Sample, a qual não encerra a interação e apenas retorna o valor corrente do DLE. A execução do programa não é suspensa e vários DLEs podem interagir simultaneamente.

3. Por evento

Neste modo, é possível a interação simultânea com diversos DLEs e cujos valores lidos podem ser aprovados pelo operador mediante uma ação de disparo.

A interação com o DLE é iniciada e finalizada por intermédio da função SetDLEMode. A leitura do valor corrente do DLE é recebido após uma ação de disparo. A ação de disparo não encerra a interação com o DLE.

Os eventos, provenientes de qualquer um dos dispositivos são inseridos em uma fila e daí podem ser retirados pelo programa de aplicação, independentemente da interação.

O operador de um dispositivo de entrada controlado pelo GKS através de um DLE, possui dois tipos de dispositivos de disparo: um de "validação" do valor de entrada, tipicamente a tecla "Enter" do teclado, e um de "invalidação", como a tecla "Break" do teclado.

5.8.9 - ESTAÇÕES DE TRABALHO

Até aqui foram descritos os recursos do GKS como um modelo independente de dispositivo pois as especificações dos periféricos gráficos não foram considerados.

Entretanto, quando as primitivas gráficas de saída precisam concretizar-se em uma imagem da tela de um terminal ou na mesa de um traçador gráfico, vários fatores específicos a cada dispositivo físico devem ser considerados.

Estes fatores específicos são suportados pelo GKS através do conceito de "estação gráfica de trabalho". Com este conceito, é possível especificar, além dos periféricos de exibição, dispositivos de entrada e periféricos para o armazenamento e transmissão de figuras.

5.8.9.1 - TIPOS DE ESTAÇÕES DE TRABALHO

Existem 6 tipos de estações de trabalho diferenciadas pelo tipo de função que suportam:

1. Estações de Saída ("output workstation")

São estações com capacidade de exibição de primitivas gráficas mas sem recursos de entrada. Como exemplo, pode-se citar o traçador de mesa.

2. Estações de Entrada ("input workstation")

São estações com apenas capacidade de entrada como é o caso da mesa digitalizadora.

3. Estações Interativas ("output-input workstation")

São estações que além de capacidade de exibição gráfica incluem recursos de entrada. A maioria dos terminais de vídeo são deste tipo, pois além da tela incluem um teclado e opcionalmente canetas de luz, ratinhos e mesas digitalizadoras.

4. Armazenador Independente de Segmentos WISS ("workstation independent segment storage")

É uma estação de trabalho "virtual" especial sem recursos de entrada ou exibição gráfica, mas com capacidade de armazenamento temporário de segmentos, possibilitando cópiar e mover seus segmentos para outras estações.

5. Metarquivo de Saída ("metafile output")

São estações de trabalho "virtuais" especiais sem recursos de entrada ou exibição gráfica, cuja função é o armazenamento por longo prazo, ou a transmissão a outras estações, de entidades geométricas.

6. Metarquivo de Entrada ("metafile input")

São estações de trabalho virtuais especiais sem recursos de entrada ou exibição gráfica, cuja função é recuperar figuras ou entidades gráficas previamente armazenadas ou transmitidas a outras estações.

As estações de entrada suportam funções para a leitura dos dispositivos de entrada, entretanto como não possuem capacidade de exibição, não incluem uma tabela de cores nem as funções que a manipulam, e são insensíveis as primitivas de saída.

Igualmente, dentro de estações de trabalho do mesmo tipo, como as estações de saída, uma instalação pode ter estações com características distintas. Por exemplo, um vídeo matricial policromático e um traçador de mesa.

Estações de trabalho com características iguais ou semelhantes compartilham uma mesma "tabela descritiva da estação" contendo os valores das constantes características de cada tipo e que podem ser consultadas pelo programa de aplicação. Nesta tabela consta por exemplo, as dimensões da superfície de exibição, se o dispositivo de saída é matricial ou vetorial, etc.

5.8.9.2 - SELEÇÃO DE UMA ESTAÇÃO DE TRABALHO

Dado que em uma instalação do GKS podem existir diversas estações de trabalho disponíveis, o programa de aplicação pode operar simultaneamente várias delas através de funções de seleção e controle.

A função `OpenWorkstation`, permite que uma estação de trabalho entre em operação (abrir a estação). Para encerrar a operação de uma estação de trabalho tem-se a função `CloseWorkstation`.

Abrir uma estação de trabalho não é suficiente para que as primitivas de trabalho sejam enviadas àquela estação. É necessário ativar a estação de trabalho através da função `ActivateWorkstation`. Para desativar a estação usa-se a função `DeactivateWorkstation`.

A desativação de uma estação de trabalho não altera a imagem sendo exibida pela estação nem os valores dos seus atributos, apenas inibe temporariamente que a estação receba as primitivas gráficas até que seja novamente ativada.

Portanto, podem-se colocar várias estações de trabalho em operação e ativar, ou desativar, uma estação ou simultaneamente várias estações durante uma sessão. Assim, por exemplo, uma primitiva gráfica pode ser enviada a um monitor de vídeo e a um traçador gráfico simultaneamente.

Para limpar a superfície de exibição de uma estação de trabalho, isto é, retirar uma imagem exibida que foi previamente criada, o GKS fornece a função `ClearWorkstation`.

Toda estação de trabalho, uma vez aberta, possui uma área de dados contendo informações sobre o estado corrente da estação, chamada "lista de estado" da estação.

As informações contidas podem ser: se a estação está ativa ou não, conteúdo da tabela de cores corrente, modo de operação dos dispositivos de entrada, etc. O programa de aplicação pode acessar estas informações através de funções de consulta.

5.8.9.3 - TRANSFORMAÇÃO DE VISUALIZAÇÃO (COORDENADAS NDC PARA DC)

O espaço normalizado NDC, onde as primitivas de saída são mapeadas pela transformação de normalização é um "espaço de exibição virtual", entretanto, o mapeamento deste espaço NDC (o quadrado $(0,1) \times (0,1)$) para as coordenadas dos dispositivos de exibição (espaço DC) não é direta pois as superfícies de exibição destes dispositivos não são quadrangulares.

As estações de trabalho possuem uma única superfície de exibição cujas dimensões podem ser obtidas pelo programa de aplicação através de uma consulta a tabela descritiva associada a cada estação. Isto não se aplica para as estações do tipo metarquivo e para a estação WISS.

O mapeamento do espaço NDC para o espaço DC é chamado de transformação de visualização ("workstation transformation"), é individual para cada estação, e é definida à semelhança da transformação de normalização por uma janela e uma vista.

Como mostra a figura (Fig. 5.16) a seguir, a janela de transformação de visualização é um retângulo com lados paralelos aos eixos coordenados do quadrado unitário do NDC.

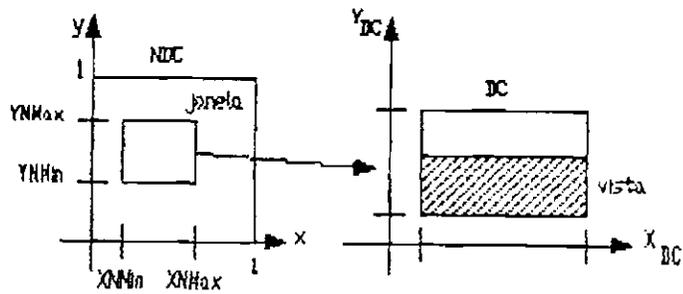


Fig. 5.16 - A janela e a vista da transformação de visualização de uma estação de trabalho.

A vista de transformação de visualização é o mapeamento da janela em um retângulo semelhante na área de exibição da estação (espaço DC).

A transformação de visualização, ao contrário da transformação de normalização, não deforma as figuras mapeadas pois utiliza o mesmo fator de escala para ambos os eixos. Pode ser distinta para cada estação de trabalho e pode ser redefinida, porém, é única em cada estação em um dado instante.

Assim, se várias estações estão ativas em um dado momento, cada uma delas pode apresentar instâncias e aspectos distintos da imagem virtual do espaço NDC. Com esta transformação, o GKS consegue produzir os efeitos de "zoom" (escala) e "pan" (vôo) da imagem.

As funções `SetWorkstationWindow` e `SetWorkstationView` do GKS, permitem definir a janela e a vista de visualização, respectivamente.

Na figura (Fig. 5.17) abaixo, é mostrada a sequência de transformações das primitivas de saída desde que são especificadas em coordenadas WCS até sua especificação em coordenadas DC.

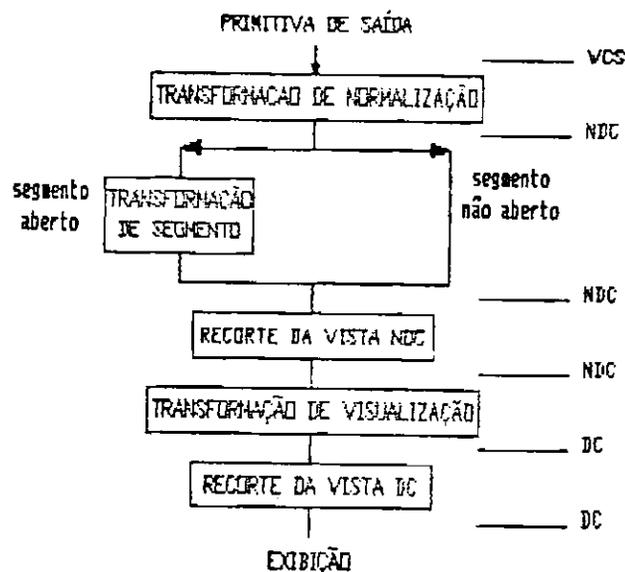


Fig. 5.17 - Sequência de transformações geométricas de coordenadas WCS a DC.

5.8.9.4 - SEGMENTOS

Alguns terminais gráficos de vídeo possuem a habilidade de armazenar e tratar segmentos. Porém, a maioria dos terminais e dispositivos físicos de exibição não possuem esta capacidade.

O GKS, com exceção das estações de trabalho de entrada e os metarquivos de entrada, simula por software a segmentação naquelas estações que não possuem esta capacidade de tal maneira que, para o usuário do GKS todas as estações de trabalho podem manipular segmentos.

Enquanto há um segmento aberto não pode-se ativar ou desativar estações de trabalho. Um segmento é removido de todas as estações ativas usando a função DeleteSegment ou apenas de uma estação em particular pela função DeleteSegmentFromWorkstation.

Todos os segmentos de uma particular estação podem ser removidos e perdidos por meio das funções ClearWorkstation e CloseWorkstation.

A mudança de atributos de um segmento afeta o segmento em todas as estações de trabalho que o contém. Assim, por exemplo, um segmento é realçado em todas as estações que o contém.

5.8.9.5 - DISPOSITIVOS DE ENTRADA

Os dispositivos lógicos da seção 5.8.7, foram descritos de uma maneira independente de estação de trabalho, entretanto, estes dispositivos estão sempre integrados a uma estação de entrada ou a uma estação interativa.

Toda estação de trabalho de entrada ou estação de trabalho interativa, possui pelo menos um dispositivo lógico de entrada DLE de cada uma das 6 classes de DLE: "locator", "stroke", "choice", "valuator", "pick" e "string".

As classes de DLE e o número de dispositivos de cada classe que uma estação de entrada ou interativa suporta, são configurados na instalação do GKS. São 6 funções do GKS, para cada classe de entrada, do tipo Inicializate(classe) que permitem inicializar um DLE a uma estação. Por exemplo, InicializateStroke.

5.8.9.6 - WISS - ARMAZENADOR DE SEGMENTOS INDEPENDENTE DE ESTAÇÃO DE TRABALHO

WISS é uma estação de trabalho especial, sem capacidade de exibição gráfica e sem capacidade de entrada, única em cada instalação do GKS, mas com capacidades potenciais de segmentação.

É um "armazenador virtual de segmentos", capaz de exportar para outras estações de trabalho segmentos nele armazenados. Portanto, pode ser considerado como um banco de segmentos do qual partes selecionadas podem ser utilizadas para exibição.

O controle do WISS é semelhante às estações de saída: pode ser aberto e fechado, enquanto aberto pode ser ativado ou desativado e enquanto ativado armazena os segmentos criados. É insensível às primitivas fora de segmentos, e os segmentos armazenados no WISS podem ser removidos da mesma forma como em outras estações.

Permite transferir primitivas gráficas de seus segmentos para outras estações. A transferência pode ser de 3 modos: cópia, associação e inserção.

1. Cópia

Neste modo, todas as primitivas armazenadas em um segmento selecionado do WISS, são enviados para uma estação de trabalho especificada, tal que a estação que as recebe as trata como primitivas fora de segmento.

As primitivas que deixam o segmento sofrem a transformação do segmento ao qual pertencem (veja seção 5.8.6.2) e o recorte da vista da transformação de normalização corrente. Ao atingirem a estação de trabalho destino, passam pela transformação de visualização ali corrente, o recorte da estação e finalmente são exibidas.

Esta operação exige que as estações WISS e a estação destino estejam ativas e não exista segmento aberto. A função CopySegmentToWorkstation, permite fazer esta cópia.

2. Associação

Neste modo, um segmento do WISS é transferido para ser armazenado em uma estação de trabalho destino. Esta operação requer que ambas as estações estejam ativas e que não exista segmento aberto. A função AssociateWithWorkstation, permite operar este modo.

3. Inserção

Neste modo, as primitivas de um segmento do WISS são copiadas a todas as estações de trabalho ativas, e estão sujeitas a uma transformação especial de inserção. Esta transformação de inserção, permite que as primitivas do segmento origem sejam posicionadas, rotacionadas ou escaladas antes de serem inseridas no segmento das estações ativas.

É necessário que haja um segmento aberto em construção nas estações destino ativas, para que o segmento do WISS seja inserido neste segmento. A função `InsertSegment`, permite operar este modo.

5.8.9.7 - METARQUIVOS

A capacidade de armazenamento de figuras em segmentos tanto em estações com recursos de exibição ou no WISS, é temporária. Ao fechar as estações de trabalho, os segmentos são destruídos.

A necessidade de armazenamento não temporário e de eventual transmissão de imagens para outras instalações, motivou a criação de uma outra estação de trabalho especial: o metarquivo.

Existem 2 tipos de metarquivos: os de saída, que armazenam ou transmitem imagens geradas pelo GKS, e os de entrada, que permitem recuperar e receber essas imagens. Ambos os tipos são considerados estações de trabalho, podem ser abertos e fechados, mas só os metarquivos de saída podem ser ativados e desativados.

Um metarquivo de saída ao ser aberto abre um arquivo (metarquivo). Ao ser ativado inicia o armazenamento sequencial em forma codificada de todas as informações gráficas contidas nas funções GKS subsequentes: primitivas gráficas de saída, seus atributos, transformações de normalização, segmentos, etc.; ao ser desativado suspende temporariamente o armazenamento sequencial até ser novamente ativado; e quando é fechado, o metarquivo é fechado.

A gravação de registros no metarquivo é totalmente transparente para o programa de aplicação, e a recuperação das informações gravadas por um metarquivo de saída é realizada através de um metarquivo de entrada.

O metarquivo de entrada ao ser aberto abre um arquivo sequencial especificado. Cada unidade de informação gravada ("item"), pode ser lida sequencialmente do metarquivo. Após ler um "item", o programa de aplicação pode submeter este "item" à interpretação por parte do GKS. A interpretação deste "item" provoca a execução da função do GKS que gerou este "item" pois cada "item" é uma função que foi gravada com determinados parâmetros.

A leitura e interpretação dos itens de um metarquivo de entrada permite reproduzir em uma estação de trabalho com capacidade de exibição a imagem gerada durante a criação do metarquivo.

A função WriteitemToGKSM do GKS, permite ao programa de aplicação gravar em um metarquivo de saída dados e informações próprias do programa.

Os metarquivos de entrada admitem 3 funções especiais: a função GetItemFromGKSM, que permite saber qual será o próximo "item" do metarquivo de entrada a ser lido, a função ReadItemFromGKSM, que efetivamente lê o próximo "item" do metarquivo de entrada, e a função InterpretItem que permite a interpretação do "item" lido e que efetivamente executa alguma função do GKS.

5.9 - GKS-3D UMA EXTENSÃO TRIDIMENSIONAL DO GKS

5.9.1 - INTRODUÇÃO

O GKS tridimensional foi projetado para proporcionar um conjunto de funções 3D com uma extensão para suportar a remoção de linhas e superfícies ocultas.

O GKS-3D é totalmente compatível com GKS de tal maneira que programas GKS podem rodar sem ser modificados no ambiente do GKS-3D. Para operar no ambiente tridimensional, o GKS-3D proporciona um conjunto completo de operações de visualização.

As características adicionadas ao GKS para torná-lo um sistema 3D, foram:

1. Primitivas 3D
2. Primitivas "fill area set" (conjunto de áreas preenchidas)
3. Processo de transformação 3D com operações de visualização ("3D transformation pipeline")

4. Processo de entrada 3D ("3D input pipeline")
5. Remoção de linhas e superfícies ocultas
6. Atributos de lados
7. Atributos geométricos 3D

5.9.2 - ESTRUTURA CONCEITUAL

GKS-3D é um sistema totalmente tridimensional, isto quer dizer que todas as operações geométricas são 3D. Conceitualmente, o modelo GKS-3D representa um único processo de transformação que suporta apenas construções 3D e trata operações 2D como subconjunto das operações 3D.

Os dados armazenados no GKS-3D tais como as primitivas nos segmentos, estão em formato 3D. Portanto, todos os dados que o GKS-3D opera são 3D mesmo que sua especificação original tenha sido 2D.

O processo de transformação está especificado de tal maneira que GKS-3D se comporte como GKS quando apenas funções 2D são usadas. Assim, programas GKS operarão corretamente no ambiente do GKS-3D mesmo que este esteja operando em 3D.

As funções GKS e GKS-3D podem ser misturadas livremente. Portanto, é possível em GKS-3D definir um objeto plano usando funções 2D do GKS e logo manipular este objeto no espaço 3D, pois as funções 2D geram primitivas 3D com a coordenada z igual a zero.

As funções 2D fornecidas para compatibilidade com GKS e para recursos de saída plana em GKS-3D, são convertidas em 3D antes de serem processadas.

5.9.3 - PRIMITIVAS DE SAÍDA

O GKS-3D transforma cada primitiva GKS ao espaço 3D e adiciona uma nova primitiva, "fill area set". As primitivas do GKS-3D são:

1. "Polyline"

Uma sequência de segmentos de linhas retas unindo uma série de pontos 3D.

2. "Polymarker"

Um símbolo centrado em um ou mais pontos 3D.

3. "Text"

Uma cadeia de caracteres em um plano especificado.

4. "Fill area"

Uma região poligonal plana que pode ser oca, preenchida com uma cor ou textura sólida, ou hachurada.

5. "Fill area set"

Uma ou mais regiões poligonais planas, cada uma das quais possui um lado. Permite especificar múltiplas áreas a serem preenchidas e o controle das aparências dos lados das faces.

Esta primitiva contém um, ou mais, conjuntos ("sets") de listas de pontos, cada um dos quais definindo um contorno em um plano.

O interior do "fill area set" é o conjunto de pontos dentro de um número ímpar de interseções. Assim, se a primitiva contém 2 conjuntos de pontos, um definindo um retângulo e o outra um círculo dentro do retângulo, o interior (a ser preenchido) seria o conjunto de pontos dentro do retângulo e fora do círculo.

6. "Cell array"

Um paralelogramo composto de células coloridas individuais em cada face.

7. "Generalized drawing primitive GDP"

Uma opção para acessar saída gráfica 3D não padronizada.

Para as primitivas que representam uma visualização plana ou possui faces planas, é associado um plano para poderem ser representadas no espaço 3D. Cada plano é definido por dois vetores 3D com os quais o GKS-3D define um "sistema de coordenadas local" para interpretar os atributos das primitivas.

5.9.4 - ATRIBUTOS

GKS-3D possui todos os atributos de GKS e mais alguns novos. Os atributos das primitivas "polyline", "polymarker" e "fill area" do GKS são os mesmos. Ao igual que em GKS, as transformações não afetam os atributos "não geométricos", por exemplo, a espessura das linhas não varia com a projeção em perspectiva.

Os atributos da primitiva "text" do GKS (seção 5.8.5.3) são os mesmos, porém a altura dos caracteres, o vetor de orientação, o alinhamento e a direção, são defini-dos no sistema de coordenadas local do plano associado a esta primitiva. A origem deste sistema é definida pela posição do texto, a direção de um dos vetores do plano como o eixo x, e a do outro vetor como o eixo y.

Os novos atributos que se aplicam a primitiva "fill area set", além dos atributos da primitiva "fill area", são:

1. "Flag edge"

Variável de controle de lado. Esta variável pode assumir o valor ligado (ON) ou desligado (OFF). Se ligado, cada área preenchida da primitiva "fill area set" possui um lado, se desligado as áreas preenchidas são desenhadas sem um lado.

2. "Edgewidth scale factor"

Fator de escala da largura do lado. Atributo com características semelhantes aos da primitiva "polyline".

3. "Edge colour index"

Índice de cor do lado. Atributo com características semelhantes aos da primitiva "polyline".

Alternativamente, qualquer um destes aspectos que controlam os lados, podem ser especificados via o atributo "Edge Index", um índice a uma "tabela de lados" dependente da estação de trabalho.

GKS-3D também permite a remoção de linhas e superfícies ocultas HLHSR ("hidden-line/hidden-surface removal"), que são funções dependentes da estação de trabalho. Neste caso, um novo atributo, o identificador HLHSR IDENTIFIER é associado a cada primitiva. A interpretação deste identificador é dependente tanto da implementação como da estação de trabalho.

A variável de modo de operação HLHSR MODE, controla se a estação de trabalho remove, ou não, as linhas e superfícies ocultas. Esta variável também depende da implementação e da estação de trabalho.

Um índice de visualização ("view index"), está associado a cada primitiva e seleciona a vista 3D correspondente em cada estação de trabalho.

5.9.5 - TRANSFORMAÇÕES E VISUALIZAÇÃO

GKS-3D proporciona 4 tipos de transformações: normalização, segmentação e inserção, visualização e estação de trabalho. Destes 4 tipos, apenas a transformação de visualização é única a GKS-3D, as outras 3 são suportadas em formato 2D por GKS.

Estas transformações juntas constituem o "processo de transformação 3D" do GKS-3D.

A figura (Fig. 5.18) abaixo, mostra estas transformações junto com os tipos de coordenadas que existem em cada estágio do processo.

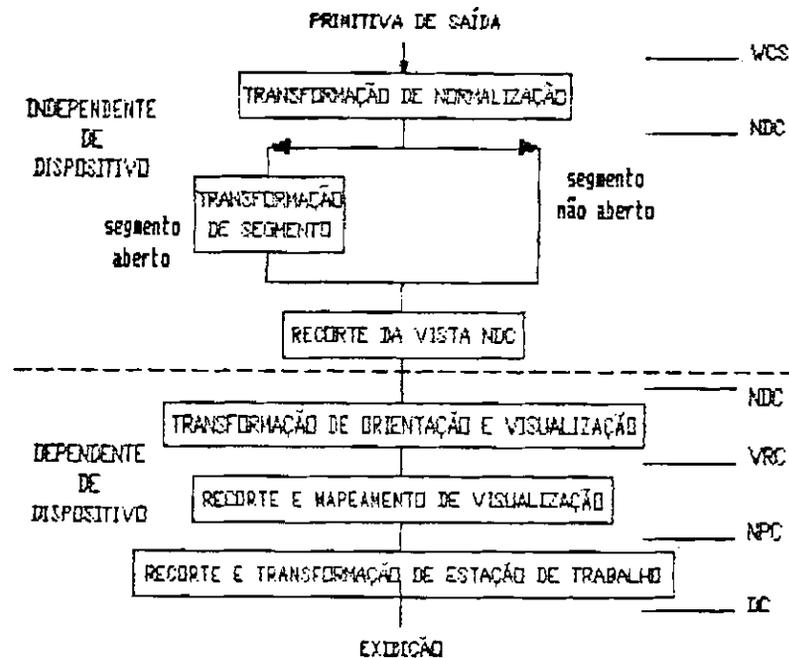


Fig. 5.18 - O processo de transformação do GKS-3D.

5.9.5.1 - TRANSFORMAÇÃO DE NORMALIZAÇÃO

A transformação de normalização do GKS-3D tem o mesmo propósito das transformações de normalização do GKS: mapeam as coordenadas globais WCS da aplicação em um espaço de coordenadas normalizadas NDC e estabelecem as fronteiras de recorte no espaço NDC.

Em GKS, as transformações de normalização são definidas por uma região retangular no espaço WCS chamada janela de normalização e uma região retangular no espaço NDC chamada vista de normalização. A transformação de normalização é o mapeamento entre estes dois retângulos.

No GKS-3D, as regiões são paralelepípedos retangulares retos. A transformação de normalização é o mapeamento entre um volume no espaço 3D em WCS e um volume em 3D no espaço NDC.

As transformações de normalização são aplicadas no momento da definição. Por exemplo, quando a função "Polyline3" é chamada, a primitiva é primeiro transformada pela transformação de normalização apropriada antes de ser posteriormente transformada.

Após a transformação de normalização, a primitiva é expressa em coordenadas NDC e é então enviada à estação de trabalho apropriada para futuro processamento de transformação.

5.9.5.2 - TRANSFORMAÇÕES DE SEGMENTO E INSERÇÃO

No GKS-3D, as transformações de segmento e inserção do espaço NDC no mesmo espaço NDC, operam da mesma forma que no GKS, exceto que as transformações são definidas por matrizes 3x4 ao invés de matrizes 2x3.

Após serem aplicadas as transformações de normalização e as de segmento e inserção a uma primitiva gráfica de saída, esta é recortada (se o recorte está habilitado) na vista de normalização associada a primitiva. O resultado é uma imagem recortada no espaço NDC.

5.9.5.3 - TRANSFORMAÇÃO DE VISUALIZAÇÃO

Os desenhos criados no espaço NDC podem ser observados usando as operações de visualização do GKS-3D.

As operações de visualização são definidas em uma "tabela de transformações de visualização" localizada em cada estação de trabalho.

No primeiro estágio de visualização, os objetos definidos no espaço NDC são transformados para o sistema de coordenadas de referência de visualização 3D VRC ("view reference coordinates") de acordo como uma matriz 4x4 de orientação de visualização. O sistema VRC é um sistema de coordenadas ortogonais definido pelos eixos n , u e v .

Qualquer transformação que possa ser expressa por uma matriz 4x4, pode ser usada para efetuar esta transformação. GKS-3D fornece funções utilitárias que podem criar estas matrizes 4x4.

O segundo estágio de visualização define mapeamentos no sistema VRC e a projeção que mapea o sistema VRC ao sistema de coordenadas de projeção normalizadas 3D NPC ("normalized projection coordinates"). Esta transformação é definida por uma matriz de mapeamento de visualização 4x4 e uma vista (viewport) 3D com um tipo de projeção que seta as fronteiras de recorte no espaço NPC.

De igual maneira, GKS-3D proporciona funções para criar uma matriz de mapeamento de visualização apropriada.

O formato do volume de visualização como mostra a figura (Fig. 5.19) abaixo, é determinado pelos dois tipos de projeção suportados por GKS-3D: paralela e perspectiva.

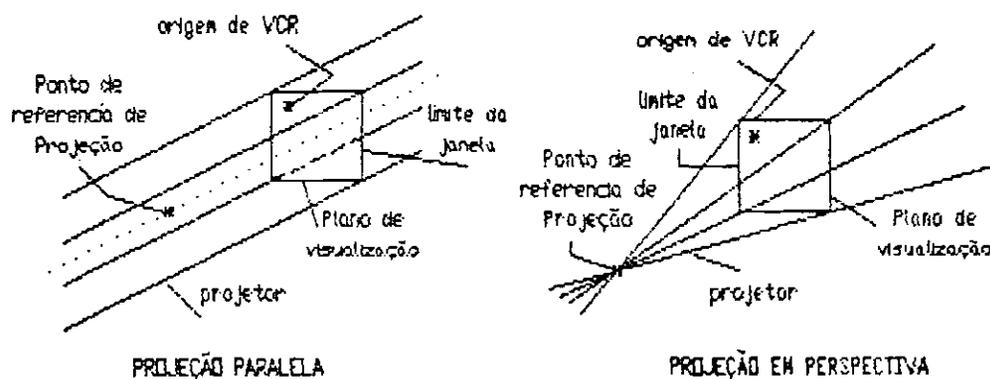


Fig. 5.19 - Volumes de visualização e tipos de projeção.

A "projeção paralela" é aquela na qual as linhas de projeção são paralelas formando um paralelepípedo que se estende ao infinito ao longo do eixo n sistema VRC.

A "projeção ortográfica" é um tipo particular da projeção paralela na qual as linhas de projeção além de serem paralelas são também perpendiculares ao plano de projeção. Esta é a projeção "default" assumida pelo GKS-3D.

Na "projeção em perspectiva", todas as linhas de projeção passam através de um único ponto, o "ponto de referência de projeção" (também chamada a posição do olho do observador), formando uma pirâmide semi-infinita com seu ápice passando pelo ponto de referência de projeção e sua base em frente do ponto de referência de projeção.

A matriz de mapeamento de visualização pode representar o mapeamento no sistema VRC com a base da pirâmide no infinito positivo (em frente ao observador) ou no infinito negativo (atrás do observador). Assim mesmo, a pirâmide pode ser truncada no seu ápice determinando uma seção frontal que pode, ou não, passar pelo ponto de referência de projeção, e uma seção posterior (base).

Conceitualmente, os volumes de visualização mostrados na figura (Fig. 5.19) anterior, são volumes infinitos ao longo do eixo n do sistema VRC. Os planos frontal e posterior, perpendiculares ao eixo n , definem o "volume de recorte de visualização".

Para a projeção paralela, o volume recortado resultante é um paralelepípedo finito. Na projeção em perspectiva, este volume pode ter a forma de uma pirâmide com dois planos onde um dos planos de recorte pode estar no ponto de referência de projeção (no caso da pirâmide não truncada).

O recorte de visualização pode ser controlado. Como resultado, o volume recortado é transformado do sistema VRC para o sistema 3D NPC (normalized projection coordinates).

5.9.5.4 - TRANSFORMAÇÃO DE ESTAÇÃO DE TRABALHO

A transformação de estação de trabalho ("workstation transformation") no GKS-3D é o mapeamento do sistema NPC para as coordenadas de dispositivo DC conceitualmente tridimensionais.

Desde que todos os dispositivos de exibição não suportam um sistema de coordenadas 3D, a componente z do sistema NPC é mapeada no plano x,y da superfície de visualização pela implementação GKS-3D.

5.9.6 - SEGMENTAÇÃO

No GKS-3D, o conteúdo do segmento consiste de primitivas 3D que foram transformadas do GKS, por transformações de normalização 3D, para o GKS-3D.

Para compatibilidade com GKS, são suportadas funções, tanto em formato 2D e 3D, para especificar as transformações de segmentos e inserção.

5.9.7 - ENTRADA

GKS-3D proporciona os mesmos tipos de entrada suportados por GKS. São incluídas as mesmas 6 classes de dispositivos lógicos de entrada DLE: localizador, cadeia de posições, identificador, avaliador, selecionador, e sequenciador.

Analogamente ao processo de visualização, todas as transformações inversas de entrada ocorrem em 3D. Assim, as classes localizador e cadeia de posições, fornecem entrada 3D real. Porém, para suportar adequadamente as transformações de visualização, estas duas classes também retornam um "índice de visualização", além do índice da transformação de normalização.

Exceto para as classes localizador e cadeia de posições, os dados retornados por cada classe são os mesmos que no GKS.

Em relação aos modos de operação destes dispositivos lógicos de entrada, o GKS-3D fornece as funções RequestLocator3, GetLocator3 e SampleLocator, além das proporcionadas por GKS.

5.9.8 - FUNÇÕES DE CONSULTA

As funções de consulta do GKS-3D são extensões lógicas das funções de consulta do GKS.

As consultas que retornam informação de dimensões ou coordenadas, são fornecidas tanto em 2D como em 3D. As consultas 2D apenas suprimem a informação da coordenada z dos valores da "lista de estado" da estação de trabalho a qual é mantida em formato 3D.

5.10 - PHIGS - "PROGRAMMER'S HIERARCHICAL INTERACTIVE GRAPHICS SYSTEM"

5.10.1 - INTRODUÇÃO

O Sistema Gráfico Interativo Hierárquico Programável PHIGS é um padrão para programação em computação gráfica. É uma norma para uma interface entre o programa de aplicação e um sistema gráfico que permite a estruturação e manipulação de dados gráficos, hierarquicamente.

PHIGS é útil para aplicações complexas que exibem dados 2D ou 3D em ambientes altamente interativos. Isto é conseguido através de uma organização hierárquica dos dados e uma flexível capacidade de edição de dados.

Como a função primária dos programas de computação é modelar ou simular sistemas reais ou hipotéticos, PHIGS foi projetado para atender as necessidades de aplicações gráficas altamente interativas e dinâmicas, tendo como objetivo liberar os programadores de aplicações da complexidade de desenvolver sistemas que exibam, e manipulem rapidamente, seus modelos aplicativos.

Projetos auxiliados por computador CAD ("Computer Aided Design"), manufatura integrada por computador CIM ("Computer Integrated Manufacturing"), processos de controle e monitoração, modelagem científica, etc., são algumas das áreas com necessidade comum de sistemas gráficos tais como PHIGS.

Todas estas aplicações possuem grandes bancos de dados organizados de diferentes maneiras; todos interagem com dados em resposta a realimentação em tempo real e/ou interação com operador; e todos refletem mudanças nos seus modelos em alguma forma de análise gráfica.

A figura (Fig. 5.20) a seguir, ilustra um sistema típico. Tal sistema mantém dois banco de dados: o "modelo da aplicação", mantido pela aplicação em uma forma apropriada para manipulado, e os "dados gráficos", mantidos pelo sistema gráfico em uma forma apropriada para análise gráfica e manipulação.

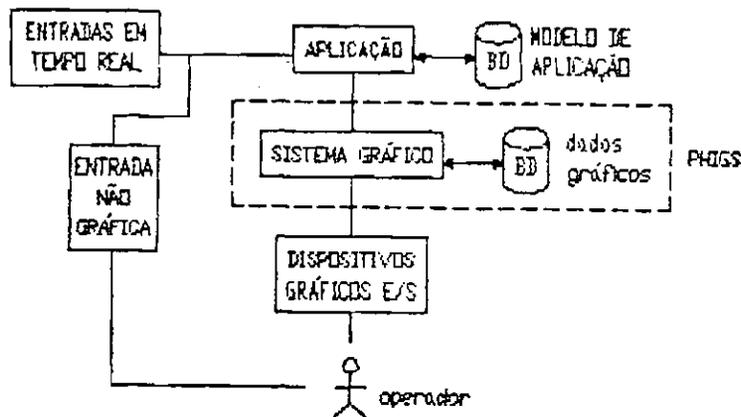


Fig. 5.20 - Um típico sistema de aplicações com gráficos.

PHIGS é capaz de descrever eficientemente o modelo de aplicação no sistema gráfico e rapidamente atualizar o modelo gráfico, atendendo a análise corrente do modelo de aplicação. A organização, conteúdo, e a capacidade de modificar os dados gráficos são aspectos que tornam PHIGS uma ferramenta para manipulação de modelos gráficos.

5.10.2 - A ORGANIZAÇÃO DO MODELO

Os modelos de aplicação são organizados segundo vários critérios, porém, independente da aplicação quase todos os modelos de aplicação são estruturados em múltiplos níveis ("multilevel"). A organização dos dados gráficos no PHIGS é também em múltiplos níveis ou hierárquica.

5.10.3 - CONTEÚDO DO MODELO

As estruturas básicas de dados gráficos no PHIGS são chamadas "estruturas". As estruturas contém os elementos gráficos padronizados encontrados em outras interfaces programáveis padrão, tais como GKS, e outras interfaces de dispositivo padrão, tais como CGI.

O poder de PHIGS não se deriva das primitivas e atributos que fornece, mas sim de como estes elementos estão organizados e são manipulados.

PHIGS poderia ter oferecido uma estrutura poderosa para um sistema gráfico baseado em um conjunto, totalmente diferente, de primitivas e atributos. Ao invés de usar primitivas e atributos que são planares e orientadas por vetores, PHIGS poderia ter substituído estas primitivas por sólidos ou volumes com atributos tais como modelos de iluminação e sombreamento.

Porém, PHIGS não evoluiu para estas primitivas e atributos, as quais estão fora dos propósitos da norma, por ainda não se ter atingido um consenso comum de conhecimento e prática. Portanto, PHIGS usa uma organização de dados gráficos e técnicas de manipulação que atualmente são amplamente aceitas e usadas.

5.10.4 - MODIFICAÇÃO DO MODELO

Além de ter uma organização de dados flexível e de múltiplos níveis, PHIGS permite a modificação dos dados gráficos em qualquer momento.

A edição dos dados gráficos permite que qualquer parte do modelo gráfico possa ser mudada, refletindo mudanças no modelo de aplicação. Esta facilidade permite uma interatividade gráfica dinâmica.

Para uma análise do modelo, PHIGS define todos os elementos primitivos geométricos e os aspectos geométricos do modelo gráfico, no que se conhece como sistema de coordenadas de modelamento ("modeling coordinates system").

A figura (Fig. 5.21) abaixo, ilustra a relação do sistema de coordenadas de modelamento com o resto do processo de transformação de PHIGS ("transformation pipeline"):

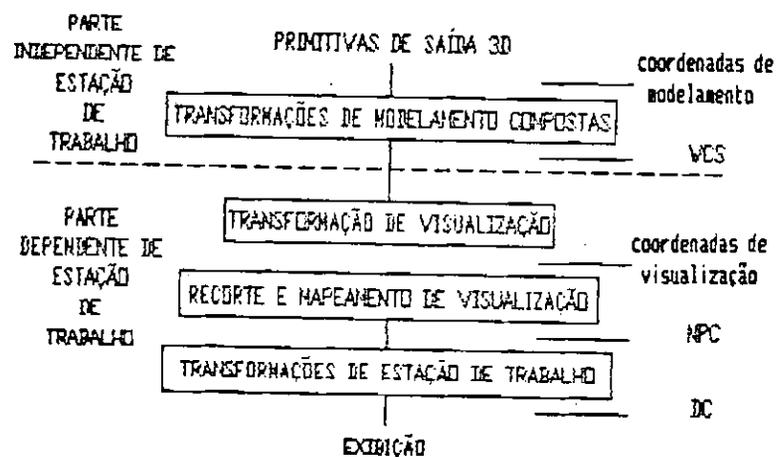


Fig. 5.21 - O processo de transformação PHIGS.

Na figura (Fig. 5.21) anterior, observa-se que os elementos do modelo gráfico podem passar por múltiplas transformações de modelamento sucessivas, cada uma mapeando os dados de volta ao espaço de coordenadas de modelamento para próximas transformações, se necessário.

É desta maneira, que as estruturas representando uma entidade geométrica são definidas. Uma entidade geométrica particular pode ser definida em um espaço de coordenadas apropriado para sua definição, e pode ser combinada com outras entidades geométricas através de transformações apropriadas.

Desde que transformações de modelamento também são elementos da estrutura de dados gráficos, e desde que cada elemento da estrutura de dados gráficos pode ser modificado a qualquer momento, efeitos interessantes e complexos podem ser obtidos.

A manipulação de um braço de robô é um exemplo clássico de modelo gráfico representado em uma hierarquia de dados modificável e flexível.

5.10.5 - A RELAÇÃO DE PHIGS COM GKS

PHIGS surge como um segundo padrão gráfico no nível de interface de programação, após GKS e sua extensão GKS-3D. Algumas das maiores diferenças entre PHIGS e GKS são:

1. Estruturas de dados

GKS proporciona uma estrutura de dados gráfica de um único nível ou plana, o "segmento". A figura (Fig. 5.22) abaixo, mostra a diferença em topologia entre os segmentos do GKS e as estruturas de PHIGS.

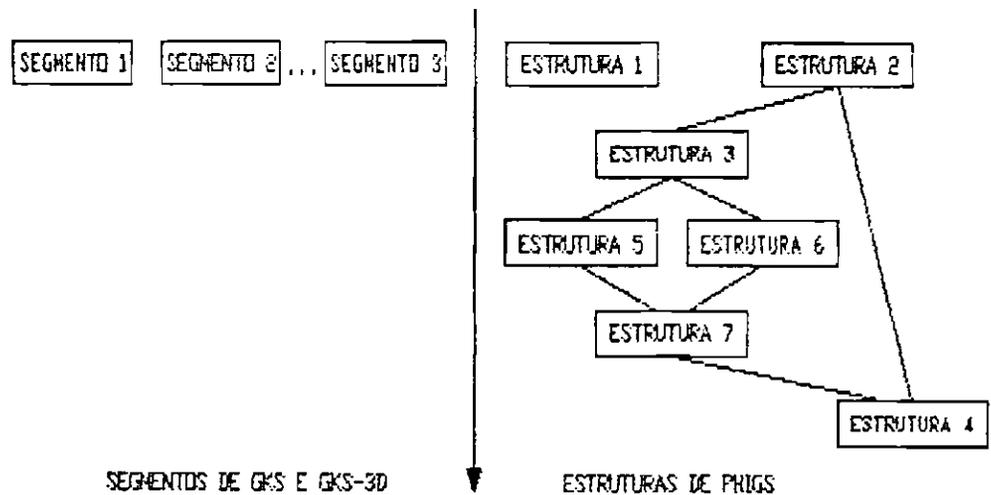


Fig. 5.22 - Comparação da topologia de dados entre GKS e PHIGS.

Os segmentos do GKS foram usados para representar informação de imagens gráficas ao invés de informação sobre um modelo gráfico.

A informação contida nos segmentos do GKS foi passada por uma normalização de coordenadas e portanto não é mais definida em termos das coordenadas do espaço onde foi criada. As estruturas de PHIGS são definidas em um espaço de coordenadas de modelamento.

2. Modificação de estrutura

Uma vez que o segmento GKS é criado, seu conteúdo não pode ser modificado. Apesar disto, certos aspectos do segmento que afetam o segmento inteiro podem ser modificados, como é o caso da visibilidade, o realce, a detetabilidade e a transformação do segmento.

Por outro lado, PHIGS permite que qualquer parte da estrutura seja modificada em qualquer instante.

Uma estrutura é formada por "elementos de estrutura", onde cada elemento de estrutura pode representar primitivas de saída, atributos, matrizes de visualização, transformações de modelamento, rótulos, nomes de conjuntos de elementos, dados, ou referências a estruturas. O programa de aplicação associa às estruturas, um único identificador para posterior referência.

3. Associação de atributos

GKS associa atributos às primitivas quando estas são criadas e antes de serem armazenadas na estrutura de dados gráfica. Para modificar os atributos das primitivas após o segmento ter sido criado, o segmento deve ser destruído e recriado com os novos atributos das primitivas.

Em PHIGS, os elementos de estrutura que representam primitivas de saída, se tornam primitivas de saída, somente quando a estrutura é percorrida (visitada) para exibição. Somente então os elementos de estrutura que representam atributos são associados às primitivas de saída.

Esta associação de atributos em tempo de percurso ("traversal time"), permite uma modificação fácil e flexível dos dados gráficos.

A figura (Fig. 5.23) abaixo, mostra as maneiras diferentes de associar atributos às primitivas de saída tanto em GKS como em PHIGS.

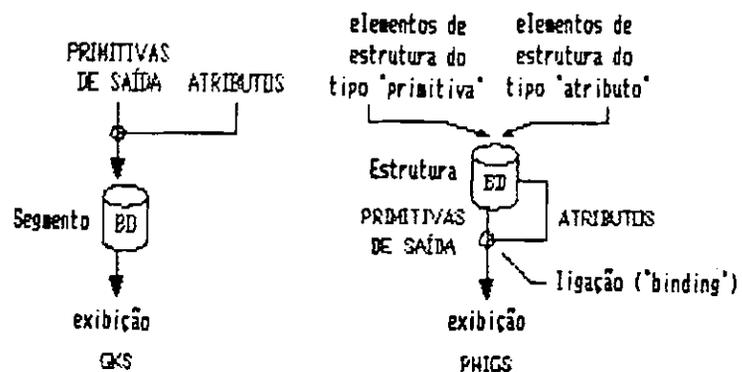


Fig. 5.23 - Comparação da associação de atributos em GKS e PHIGS.

5.10.6 - ÁREAS FUNCIONAIS DE PHIGS

Como descrito, os dados gráficos em PHIGS estão organizados em estruturas, onde cada estrutura contém elementos de estruturas. Esta organização, controlada pelo programa de aplicação, permite ao programador arranjar os dados da melhor maneira conveniente.

5.10.6.1 - HIERARQUIA DE ESTRUTURAS

Uma hierarquia de estruturas é criada através do uso do elemento de estrutura EXECUTE. Durante o percurso da estrutura para exibição, este elemento permite que outra estrutura seja executada dentro da estrutura original, resultando em uma organização hierárquica, como mostra a figura (Fig. 5.24) abaixo.

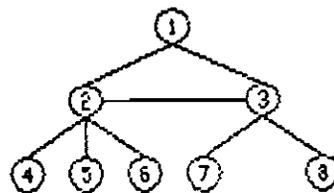


Fig. 5.24 - A organização de dados hierárquica em PHIGS.

Na figura (Fig. 5.24) anterior, a estrutura 1 contém elementos de estrutura EXECUTE, referenciando as estruturas 2 e 3. A estrutura 1 é chamada de pai das estruturas 2 e 3. As estruturas 2 e 3, são chamadas filhos da estrutura 1.

Não existe limite no número de estruturas que podem executar uma dada estrutura, nem há limite na profundidade da hierarquia.

Para que uma hierarquia de estruturas seja percorrida para exibição, uma estrutura deve ser associada a uma estação de trabalho (no exemplo da figura (Fig. 5.24) anterior, a estrutura 1 seria associada). O percurso das estruturas se inicia nas estruturas associadas às estações de trabalho.

Porém, PHIGS permite criar estruturas que não são executadas como parte de outras estruturas, nem associadas a uma estação de trabalho.

PHIGS armazena estas estruturas para serem depois usadas em elementos de estruturas EXECUTE ou como estruturas associadas a estações de trabalho, para uso posterior da aplicação. Um exemplo é uma biblioteca de símbolos composta de uma coleção de estruturas.

5.10.6.2 - ASSOCIAÇÃO DE ATRIBUTOS EM TEMPO DE PERCURSO

Assim como outros sistemas gráficos, PHIGS usa atributos para definir a aparência de uma primitiva de saída. Os atributos definidos em PHIGS são aqueles mesmos definidos em outros padrões gráficos como GKS ou CGI, porém em PHIGS estes são associados no tempo de percurso da estrutura e não quando as estruturas são criadas.

Como a atribuição de atributos é modal (assume os valores correntes), quando um elemento de estrutura EXECUTE é encontrado durante o percurso da estrutura, ocorre o seguinte:

1. O percurso da estrutura corrente é suspensa
2. Os valores dos atributos são salvos
3. A estrutura executada (e todas as estruturas que este executa) é percorrida
4. Os valores dos atributos salvos são restaurados ao estado antes da estrutura executada ser percorrida
5. O percurso da estrutura pai continua

Portanto, uma estrutura subordinada (filho) herda os valores dos atributos do seu pai em vigor quando este foi executado. Este processo é definido como "associação de atributos em tempo de percurso".

Assim, como os valores dos atributos são restaurados ao estado prévio antes do percurso da estrutura, uma estrutura afeta somente as estruturas subordinadas a esta. As estruturas associadas às estações de trabalho, herdam seus atributos iniciais de um estado "default" global.

5.10.6.3 - NOMES DE CONJUNTOS E FILTROS

As aplicações geralmente precisam mudar a visibilidade, o realce, ou a detetabilidade de conjuntos relacionados de primitivas de saída.

O agrupamento de primitivas de saída em conjuntos é específica da aplicação. Frequentemente, todas estas primitivas podem não ser elementos da mesma estrutura ou estar diretamente relacionadas na hierarquia da estrutura. Além disso, uma primitiva em particular pode pertencer a mais de um conjunto.

PHIGS fornece um método flexível para manipular tais situações com o atributo NAME SET (nome de um conjunto de primitivas agrupadas) e filtros que permitem filtrar primitivas dos conjuntos das operações de percurso no que diz respeito à visibilidade, realce e detetabilidade.

5.10.6.4 - PROCESSO DE TRANSFORMAÇÃO

PHIGS foi projetado para descrever objetos 3D, orientá-los no espaço, descrever seus movimentos, e visualizar os resultados. Para isto, são fornecidos recursos para a descrição das transformações e suas modificações dinâmicas.

As transformações fornecidas e suas relações, como mostrado na figura (Fig. 5.21) anterior, formam o processo de transformação de PHIGS ("transformation pipeline").

1. Transformações de modelamento

As transformações de modelamento são usadas para localizar e orientar objetos no espaço e em relação com outros objetos.

Estas transformações convertem o espaço de coordenadas de modelamento das primitivas de saída ao espaço de coordenadas universais WCS. Os elementos de estruturas descrevem as transformações de modelamento como matrizes 4x4, portanto estas transformações são armazenadas como elementos de estruturas.

Funções utilitárias são fornecidas para definir matrizes para as operações de escala, translação e rotação. A transformação de modelamento pode ser substituída, ou uma nova transformação de modelamento pode ser concatenada através de composição de matrizes com a transformação de modelamento prévia.

2. Transformações de visualização

As transformações de visualização descrevem a orientação da vista da cena. Estas transformações mapeiam o espaço de coordenadas universais WCS da transformação de modelamento ao sistema de coordenadas de visualização VCS.

As transformações de visualização são descritas como matrizes 4x4 em uma "tabela de visualizações" da estação de trabalho.

Funções utilitárias são fornecidas para suportar um modelo de visualização 3D, mas outros modelos de orientação de visualização podem ser construídos alterando diretamente os elementos da especificação da matriz de visualização 4x4.

3. Mapeamento de visualização e recorte

A parte de mapeamento de visualização e recorte do processo de transformação, descreve o mapeamento do sistema de coordenadas de visualização VCS para o espaço de coordenadas de projeção normalizadas NPC ("normalized projection coordinates").

Estas transformações de janela para vista são armazenadas na tabela de visualizações da estação de trabalho, não como elementos de estruturas.

Os componentes destas transformações são a distância ao plano de visualização, a janela, os planos de recorte frontal e posterior, o ponto de referência de projeção, a vista 3D, os indicadores de recorte, e se a projeção é paralela ou em perspectiva (seção 5.9.5.3).

4. Transformações de estação de trabalho

A transformação de estação de trabalho mapeia o espaço NPC para o espaço de coordenadas de dispositivo DC de uma estação de trabalho.

Esta transformação permite selecionar um volume no espaço NPC (a janela da estação de trabalho) e o mapeia em um volume no espaço de coordenadas de dispositivo (vista da estação de trabalho). Isto permite selecionar a posição de uma imagem dentro do espaço DC da estação de trabalho.

A especificação das transformações de modelamento e a seleção das entradas da tabela de visualizações da estação de trabalho, se comportam como qualquer outro atributo de PHIGS. Por esta razão, seus valores são salvos e restaurados na entrada e saída de estruturas durante o percurso da hierarquia.

O processo de transformação em PHIGS é conceitualmente um processo 3D. PHIGS permite operações 2D, porém são operações 3D com a coordenada do plano z igual a 0, o que equivale a desenhar em um plano no sistema de coordenadas 3D. A orientação deste plano no espaço de coordenadas universais 3D é controlada pelas transformações de modelamento.

5.10.6.5 - EDIÇÃO DE ESTRUTURAS

A interação dinâmica das aplicações, fornecida por PHIGS, é possível graças a capacidade de modificar o conteúdo das estruturas através da edição das estruturas.

Isto permite a edição dos atributos, mudar as primitivas de saída, e modificar a hierarquia através de uma técnica simples: todos os elementos de estrutura, sejam eles atributos, primitivas, ou elementos de execução de estruturas, podem ser modificados, inseridos, ou apagados usando edição de estruturas.

Para editar uma estrutura, esta é aberta e o ponteiro dos elementos da estrutura é posicionado no último elemento da estrutura. Novos elementos são inseridos após o último elemento, o ponteiro pode ser movido para um rótulo de um elemento da estrutura, ou o ponteiro pode ser movido para um número de elemento de estrutura específico.

Elementos podem ser apagados ou adicionados no início, meio ou fim da estrutura, sem restrição no seu número, tipo ou localização.

A criação de uma estrutura é outra forma de edição de estrutura. Se a estrutura não existe, ao abrir uma, esta é automaticamente criada e o ponteiro se posiciona no final da estrutura (início), pronto para receber elementos.

5.11 - CONCLUSÃO DO ESTUDO SOBRE PADRONIZAÇÃO GRÁFICA

Em relação a padronização gráfica, o modelador geométrico implantado, conceitualmente faz parte do modelo de referência para padronização gráfica (seção 5.4), no sentido de realmente ser um programa de aplicação que representa uma interface entre o usuário (programador de aplicações) e um pacote de suporte de computação gráfica, no caso o pacote gráfico CADDs 4X, usado na implementação.

O pacote gráfico usado (CADDs 4X), interage com os dispositivos gráficos físicos através de funções de entrada e saída já implementadas.

A figura (Fig. 5.25) mostrada abaixo, descreve o modelador geométrico implantado dentro do modelo de referência para padronização gráfica.

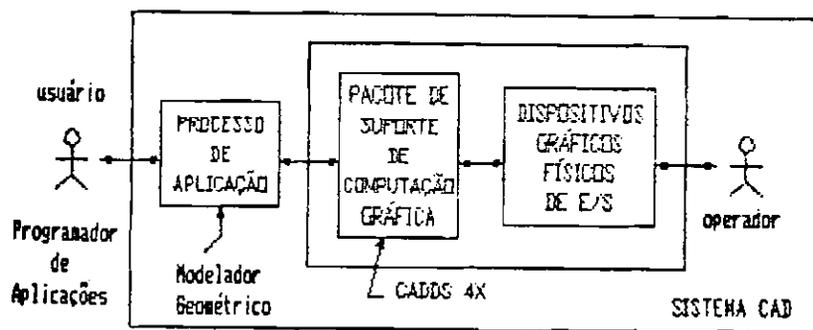


Fig. 5.25 - O modelador geométrico dentro do modelo de referência da padronização gráfica.

Na figura (Fig. 5.25) anterior, o programador de aplicações é o projetista usuário do sistema CAD que usa o programa de aplicação desenvolvido, para modelar sólidos.

Este programa de aplicação usa as rotinas gráficas já implementadas, porém não padronizadas, que o pacote gráfico CADD5 4X oferece.

Da mesma maneira, o pacote gráfico CADD5 4X, já possui funções internas que controlam os dispositivos de entrada, especificamente a caneta de luz com mesa digitalizadora e teclado, disponíveis no sistema, e funções que controlam os dispositivos de saída tais como, terminal de vídeo, impressora e "plotter", também disponíveis.

O operador é o gerente do sistema, responsável pelo funcionamento e manutenção do sistema.

Da mesma maneira como foi usado o pacote gráfico CADD5 4X, seria desejável usar outros pacotes gráficos padronizados tais como GKS, GKS-3D ou PHIGS, para implementar o modelador geométrico, com o objetivo de alcançar a transportabilidade do software desenvolvido.

Porém, esta transportabilidade altamente desejável, teria um custo de desenvolvimento adicional significativo, em razão de que estes pacotes gráficos padronizados não suportam, por não ser o objetivo da padronização, operações gráficas complexas como a varredura tanto rotacional como translacional, o sombreado e principalmente as operações booleanas, as quais são oferecidas pelo pacote gráfico CADD5 4X.

Estas características constituem as ferramentas vitais e imprescindíveis no projeto do modelador geométrico desenvolvido, pelos seguintes motivos:

- a varredura rotacional e translacional, são necessárias para gerar o sólido primitivo ou nível 1 da estrutura de dados apresentada, o que também implica em desenvolver rotinas de visualização em 3D e de definição de planos de construção, embora já presentes nos pacotes 3D;
- as operações booleanas necessárias para gerar os sólidos combinados e o sólido final, níveis 2 e 3 da estrutura de dados, através das operações de união, intersecção e diferença entre sólidos;
- o sombreado que permite visualizar de um forma real o sólido gerado, evitando a ambiguidade do sólido visto no seu modelo em arame,

Sem a disponibilidade destas operações gráficas dentro dos pacotes gráficos padronizados, o software implementado ficaria limitado a apenas poder gerar os contornos 2D usados para a operação de varredura na geração dos sólidos primitivos, isto é, só poderiam ser geradas as folhas da árvore ou nível 0 da estrutura de dados apresentada.

Portanto, o software que implementa o modelador geométrico teria que ser estendido para incluir novas rotinas gráficas que implementem a varredura, as operações booleanas e o sombreado usando as primitivas padronizadas destes pacotes gráficos padronizados como o caso do GKS-3D e o PHIGS.

Este desenvolvimento pode ser possível, e até mesmo otimizado e facilitado, graças às ligações destes pacotes gráficos com linguagens de alto nível, amplamente usadas hoje em dia e estruturadas e compiladas, e também poderosas em comparação com a linguagem VARPRO 2. A linguagem VARPRO 2 por ser uma linguagem interpretada, embora estruturada, não permite definir estruturas dinâmicas, o que dificulta operações em árvore típicas nestas aplicações. Este desenvolvimento seria muito caro, pela complexidade da geometria matemática por trás dos algoritmos a serem implementados.

Entretanto, uma grande vantagem do uso de um dos pacotes gráficos padronizados na implementação, é que conseguindo-se a transportabilidade do modelador geométrico, este poderia ser usado em estações de trabalho com tempo de processamento muito menor. A velocidade de processamento do sistema utilizado, embora o fato de que as operações gráficas complexas que oferece seja um fator predominante, poderia ser incrementada se o processador central que cuida da atividade multiusuária e multitarefa fosse mais rápido, pois atualmente é um processador de apenas 16 bits.

O fato do pacote gráfico CADDS 4X oferecer funções gráficas bastante complexas tais como visualização em 3D, definição de planos, operações booleanas e sombreamento, etc., as quais são usadas pelo modelador gráfico implementado, dificultaria sua implementação no pacote gráfico GKS por ser um pacote 2D.

GKS-3D, a extensão do GKS, e PHIGS, seriam mais adequados para este objetivo, por serem pacotes gráficos tridimensionais. Porém, PHIGS seria mais aconselhável para este tipo de implementação em particular, pois em função da manipulação dinâmica das estruturas de dados e a associação de atributos em tempo de visualização, tornaria a aplicação mais rápida e fácil de ser alterada.

O padrão de interface com dispositivos gráficos CGI, assim como o padrão de formato e transferência de arquivos CGM, estão em um nível de implementação menor que o programa de aplicação implementado.

CGI e CGM, são transparentes para o modelador geométrico que usa um pacote gráfico, pois estão implementados dentro do nível interno do sistema CAD. O sistema CAD utilizado não usa este tipo de padrões, o que dificulta principalmente a transportabilidade dos desenhos existentes.

CAPÍTULO 6

AVALIAÇÃO DE RESULTADOS E OBJETIVOS ALCANÇADOS

6.1 - INTRODUÇÃO

A seguir é apresentada uma avaliação dos resultados e objetivos alcançados na implantação do modelador geométrico no sistema CAD utilizado.

Nesta avaliação são explicadas as dificuldades encontradas durante a implementação, as várias etapas do desenvolvimento do trabalho e discute-se com profundidade a interatividade do software com o usuário, apresentando o fluxograma de cada menu de interação.

6.2 - ETAPAS DE DESENVOLVIMENTO

Inicialmente, após a proposta do trabalho, foi iniciada uma aprendizagem prática com o sistema, para se ter uma familiarização com os comandos gráficos e poder usar os recursos que estes oferecem.

A familiarização com o sistema, consistiu de testes de todas as entidades gráficas, identificando e interpretando seus parâmetros associados.

Seguindo com a interpretação do ambiente gráfico de trabalho relacionado com os conceitos de arquivos de trabalho, folhas de desenho, grades, unidades de medida, níveis de trabalho, espaços 2D e 3D, vistas ortográficas e isométrica, definição de planos de construção, e sistemas de coordenadas.

Dado o interesse de construir sólidos por varredura, foram exploradas e testadas todas as funções da biblioteca relacionadas com varredura, tanto rotacional como translacional. A partir dos sólidos resultantes, seguia uma análise da aparência dos sólidos, seu volume ocupado, sua posição dentro do espaço do sistema de coordenadas, a interpretação da representação em arame, e de como visualizar o sólido de diferentes vistas.

A influência dos planos de construção nos resultados assim como as vistas relacionadas com estes planos, dificultaram na interpretação dos resultados.

Para os sólidos terem uma forma realista, foi utilizado o sombreamento. Foi abordado, como pintar um sólido, quais cores disponíveis, como determinar a cor do fundo, qual a interpretação do ponto de iluminação, o motivo do sólido apresentar sombras em uma determinada direção e descontinuidade na cor. Assim mesmo, onde localizar o foco de iluminação, como mudá-lo de posição e interpretar a pintura resultante. Portanto, os parâmetros da rotina de sombreamento foram exaustivamente testados e os resultados interpretados.

Com relação às operações booleanas, foram testadas as operações de união, intersecção e diferença entre dois sólidos, e interpretados os sólidos resultantes.

Não foi suficiente somente a familiarização dos comandos gráficos a nível do pacote gráfico CADD5 4X; foi também necessário estudar o sistema operacional OS, as rotinas de gerenciamento de arquivos, estruturas de diretórios, comandos próprios do sistema operacional e o uso de editores de texto.

Para programar manipulando todas as rotinas gráficas, a linguagem gráfica VARPRO 2 foi utilizada. Foram abordados: sua sintaxe, os tipos de dados suportados, a definição das variáveis, as estruturas de controle, as estruturas de armazenamento, a manipulação de arquivos, a programação do sistema operacional dentro da linguagem VARPRO 2, as chamadas de rotinas gráficas de CADD5 4X, e a execução de um programa.

Todos estes itens citados foram utilizados, programando e entendendo esta linguagem.

Paralelamente foi iniciado um estudo teórico sobre banco de dados, modelagem geométrica, esquemas de representação de sólidos, e padronização gráfica.

Em virtude do objetivo do trabalho ser a implantação de um modelador geométrico com estrutura de dados hierárquica que manipula sólidos gerados por varredura, seguindo uma estrutura em árvore binária, onde os nós representam operações booleanas ou de transformação, foi necessário fazer, em paralelo à familiarização do sistema, um estudo teórico para fundamentar o que se entende por banco de dados e a partir do estudo das diferentes abordagens de estruturas de dados, poder partir para um modelo próprio aplicável a um sistema gráfico.

O modelamento geométrico também teve que ser estudado em função da estrutura de dados do modelador geométrico implantado, manipular entidades gráficas, das quais deve-se conhecer como estas são construídas, quais os parâmetros que as definem, e qual é o fundamento matemático nas quais estão apoiadas.

A necessidade de representação de um sólido complexo a partir de sólidos mais simples, determinou a pesquisa de todos os esquemas de representação de um sólido para poder ser determinado quais deles poderiam ser combinados para resolver adequadamente a representação dos sólidos na estrutura de dados, armazenada no banco de dados gerado.

Finalmente, foi realizado um estudo sobre Padronização Gráfica, para analisar a possibilidade do modelador geométrico ser implantado em outro sistema, usando um pacote gráfico padronizado, diferente do pacote gráfico não padronizado, utilizado no trabalho.

Assim, uma vez conhecidas as ferramentas gráficas disponíveis e determinada a estrutura de dados a ser implementada, partiu-se para o desenvolvimento de um software que implemente esta estrutura de dados e atenda a todas as características descritas na proposta apresentada no capítulo 2 deste trabalho.

O software está escrito na linguagem VARPRO 2, e representa um programa de aplicação, o qual é uma interface de alto nível entre o projetista e toda a complexidade do sistema.

É um software amigável e está implementado através de menus de tela, o que o torna bastante inteligível, e, de uma maneira sequencial, permite ao projetista modelar hierarquicamente qualquer sólido a partir de sólidos mais simples.

Todos os dados solicitados ao usuário são entrados via teclado ou via caneta eletrônica.

O software suporta as operações de varredura para a construção dos sólidos primitivos, as operações booleanas e de transformação, e o sombreado.

Como suporte a estas ferramentas também é possível definir folhas de desenho, vistas, níveis de trabalho e planos de construção.

Todas as entidades 2D selecionadas usadas para definir os contornos fechados, também estão disponíveis. Um utilitário exclusivo, para este tipo de entidades, também foi projetado.

É um software inteligente, pois conforme o projetista vai gerando e compondo os sólidos, o software monta uma estrutura de dados dinâmica onde todo tipo de informação e operação é armazenado. A estrutura de dados (listas dinâmicas) fica na memória enquanto o arquivo de trabalho está ativo, para logo ser armazenada em um banco de dados (disco) após desativar o arquivo de trabalho.

Assim, se o usuário desejar dar continuidade a um arquivo de trabalho por ele já construído, ao se ativar este arquivo de trabalho, toda a informação gráfica (o modelo) da estrutura de dados armazenada no banco de dados, é recuperada do disco para a memória.

Após a recuperação do arquivo de trabalho, o usuário pode modificá-lo e o software continua implementando o banco de dados a partir do existente.

Outra dificuldade enfrentada pelo software na construção desta estrutura de dados hierárquica, foi a estrutura de armazenamento não dinâmica da linguagem VARPRO 2. Assim, toda a estrutura dinâmica teve que ser simulada a partir da estrutura estática desta linguagem. Os ponteiros foram simulados e as listas dinâmicas implementadas.

6.3 - FLUXOGRAMAS DO MODELADOR GEOMÉTRICO

Nas figuras seguintes serão mostrados todos os fluxogramas que descrevem a implementação e operação do modelador geométrico no sistema. E ainda, os itens a seguir descrevem cada fluxograma.

6.3.1 - FLUXOGRAMA "MENU PRINCIPAL"

Este fluxograma está mostrado na figura (Fig. 6.1) a seguir.

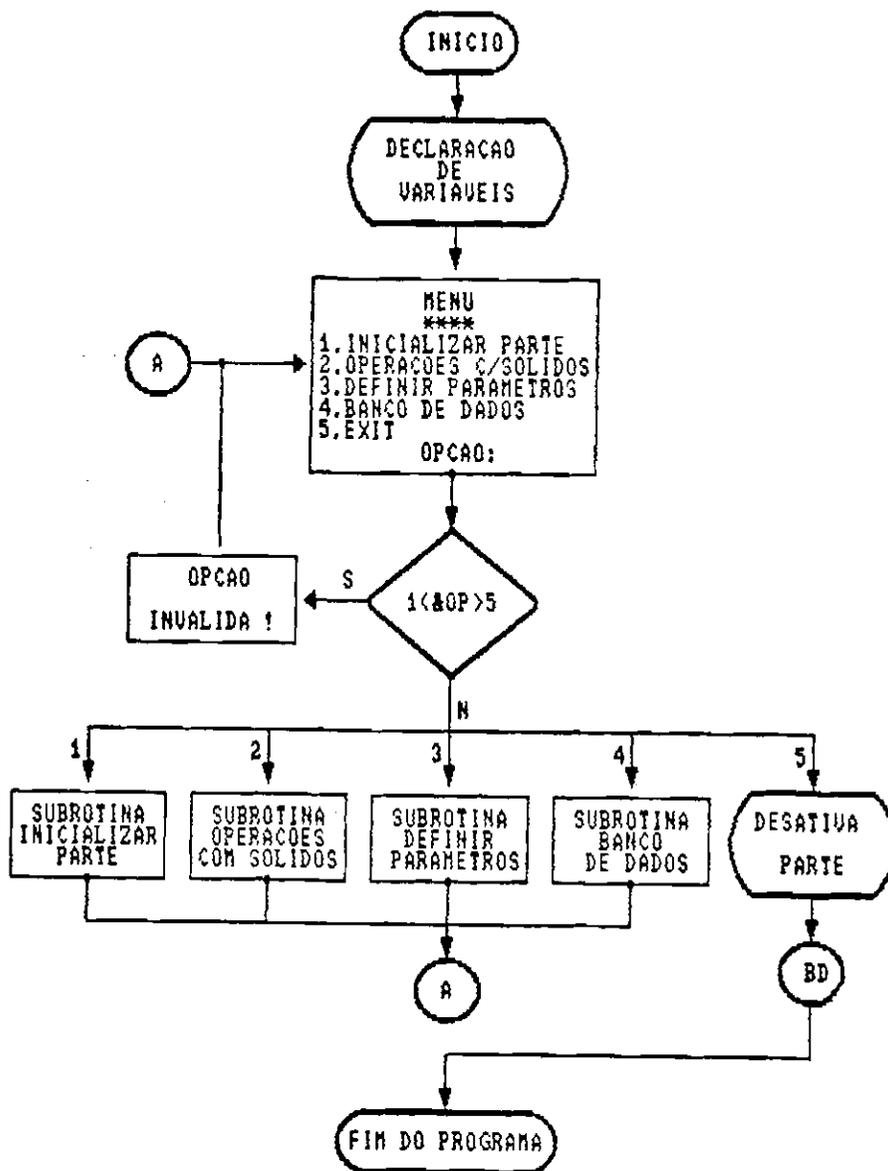


Fig. 6.1 - Fluxograma "menu principal".

Inicialmente são declaradas todas as variáveis do programa e é apresentado o menu onde se pode escolher:

1. Inicializar uma "parte"

Abre um arquivo gráfico de trabalho no nível CADDs.

2. Operações com sólidos

Para modelar um sólido hierarquicamente.

3. Definir parâmetros

Determina o ambiente gráfico a ser usado, permitindo selecionar folhas de desenho, vistas, planos de construção, etc.

4. Banco de dados

Permite operações na estrutura de dados armazenada no banco de dados.

5. Exit

Abandona o programa, dando opção de salvar o arquivo de trabalho e o banco de dados em disco.

6.3.2 - FLUXOGRAMA "SUB-ROTINA INICIALIZA PARTE"

Este fluxograma está mostrado na figura (Fig. 6.2) a seguir.

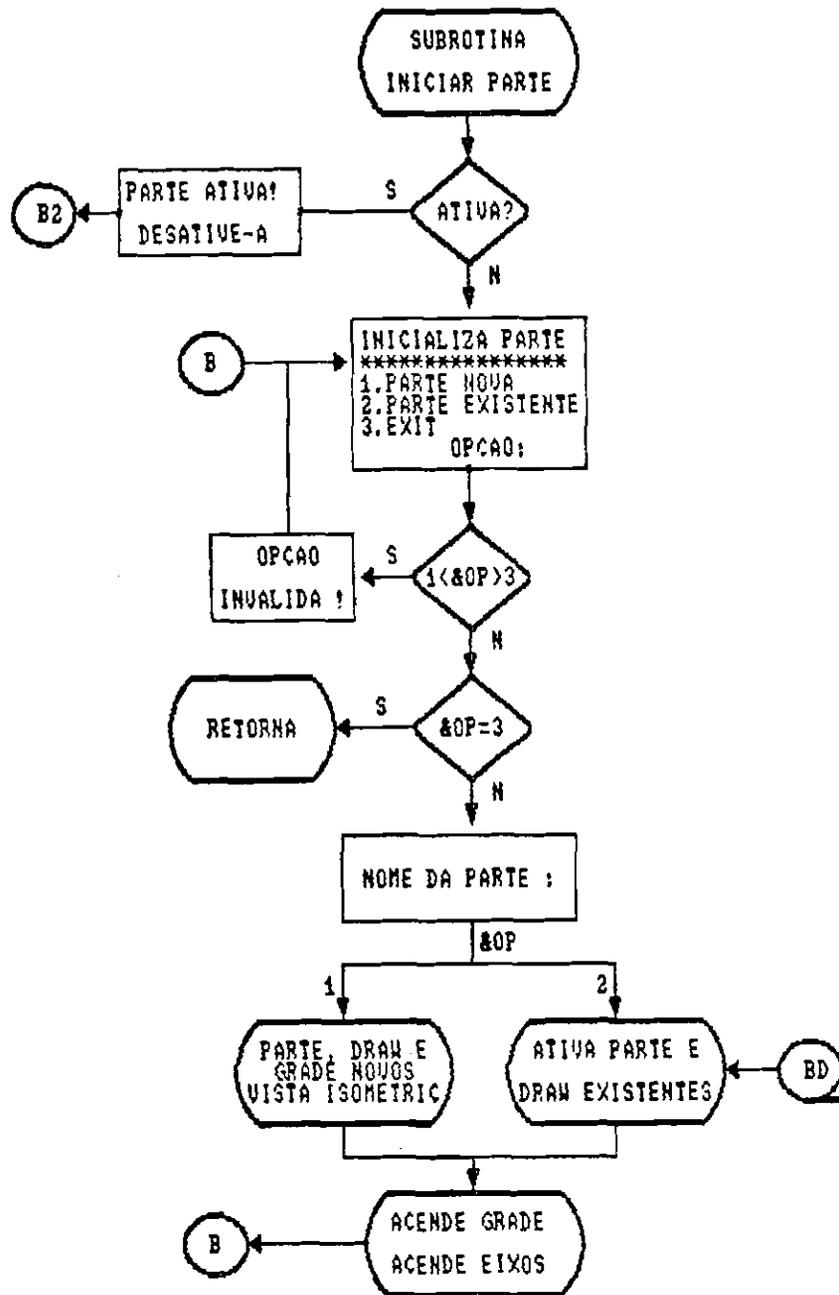


Fig. 6.2 - Fluxograma "sub-rotina inicializa parte".

Esta sub-rotina inicializa um arquivo de trabalho ("parte") no sistema. Se já existe um arquivo de trabalho ativo, a subrotina retorna pois o usuário deverá primeiro desativa-lo, porque só se pode ativar um arquivo de trabalho por vez.

Neste menu pode-se escolher:

1. Parte nova

Ativa um novo arquivo de trabalho, onde por "default", o programa solicita o nome do arquivo, define milímetros como unidade de medida, tamanho A3 para a folha de desenho, plano de construção de topo, vista isométrica, distância entre os pontos da grade igual a 5, e liga a grade e os eixos de coordenadas.

2. Parte existente

Ativa um arquivo de trabalho que já foi criado. O arquivo é carregado com todos seus parâmetros de definição de ambiente gráfico originais. São solicitados os nomes do arquivo de trabalho e da folha de desenho. Ao ativar-se a folha de desenho, aparecem o modelo e as vistas associadas, e o banco de dados é recuperado do disco.

3. Exit

Retorna ao menu principal.

6.3.3 - FLUXOGRAMA "SUB-ROTINA OPERAÇÕES COM SÓLIDOS"

Este fluxograma está mostrado na figura (Fig. 6.3) a seguir.

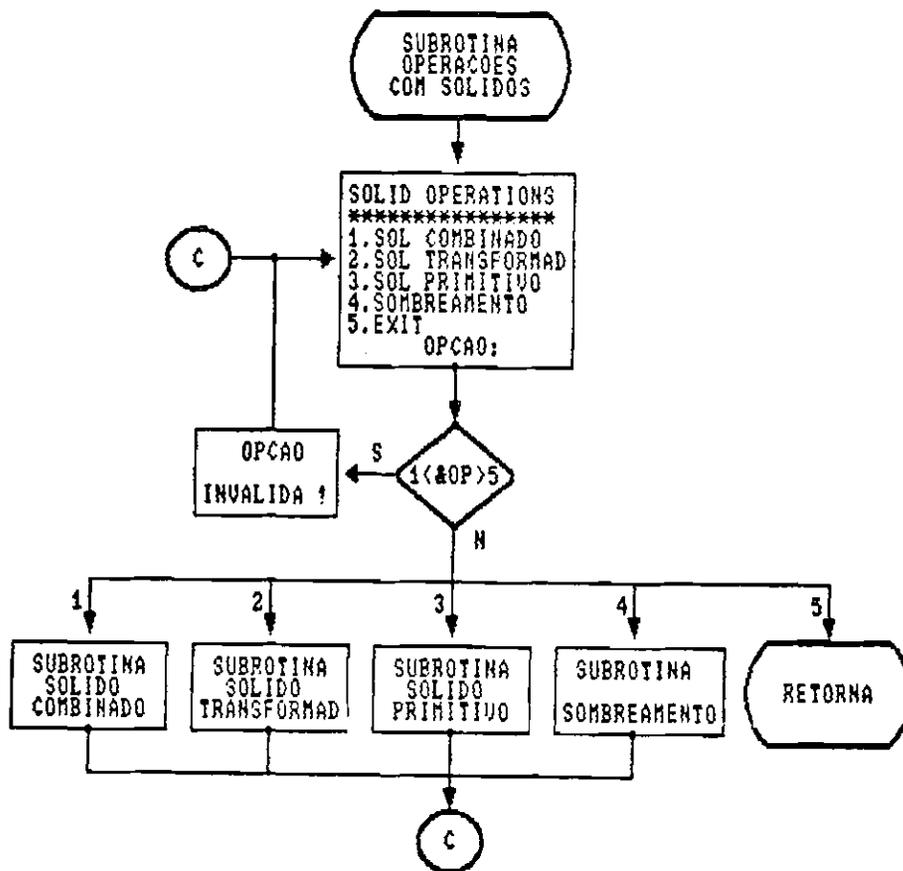


Fig. 6.3 - Fluxograma "sub-rotina operações com sólidos".

Esta sub-rotina permite operações com sólidos. O menu é apresentado com as seguintes opções:

1. Sólido combinado

Permite combinar dois sólidos usando operações booleanas.

2. Sólido transformado

Permite usar as operações de transformação de corpo rígido em sólidos.

3. Sólido primitivo

Usado para construir um sólido primitivo.

4. Sombreamento

Permite dar cor ao modelo de um sólido, assim como visualizar outros modelos.

5. Exit

Retorna ao menu principal.

6.3.4 - FLUXOGRAMA "SUB-ROTINA SÓLIDO COMBINADO"

Este fluxograma e sua continuação, estão mostrados nas figuras (Fig. 6.4) e (Fig. 6.5) a seguir.

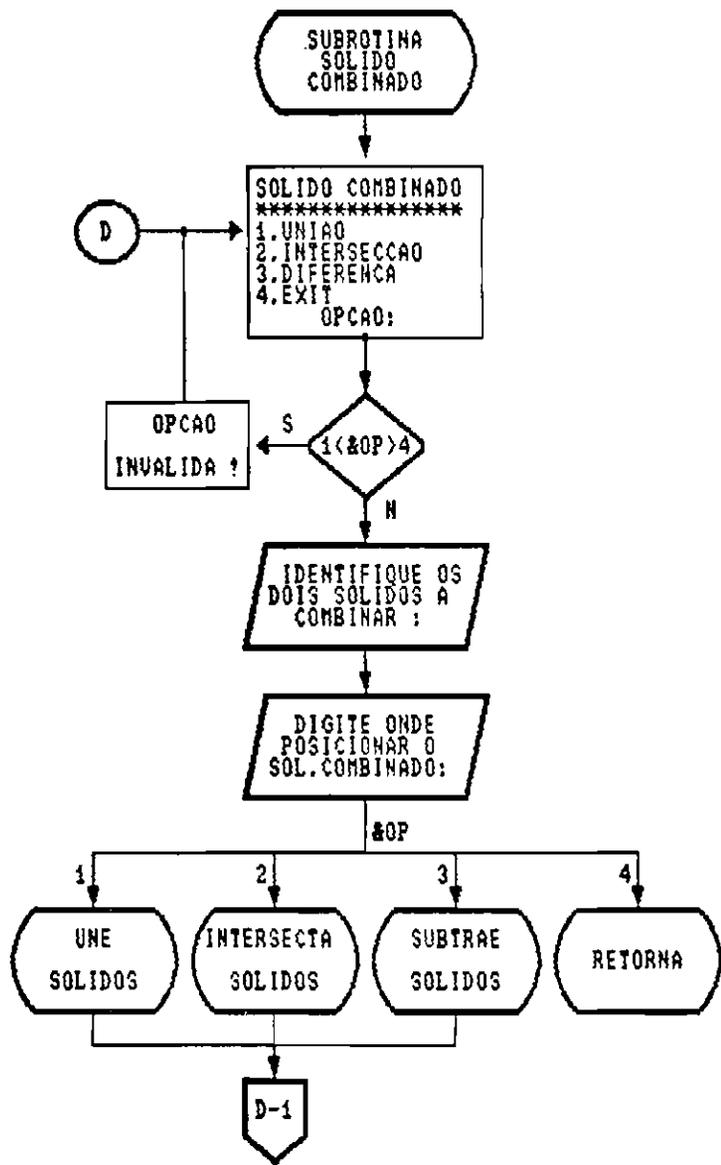


Fig. 6.4 - Fluxograma "sub-rotina sólido combinado".

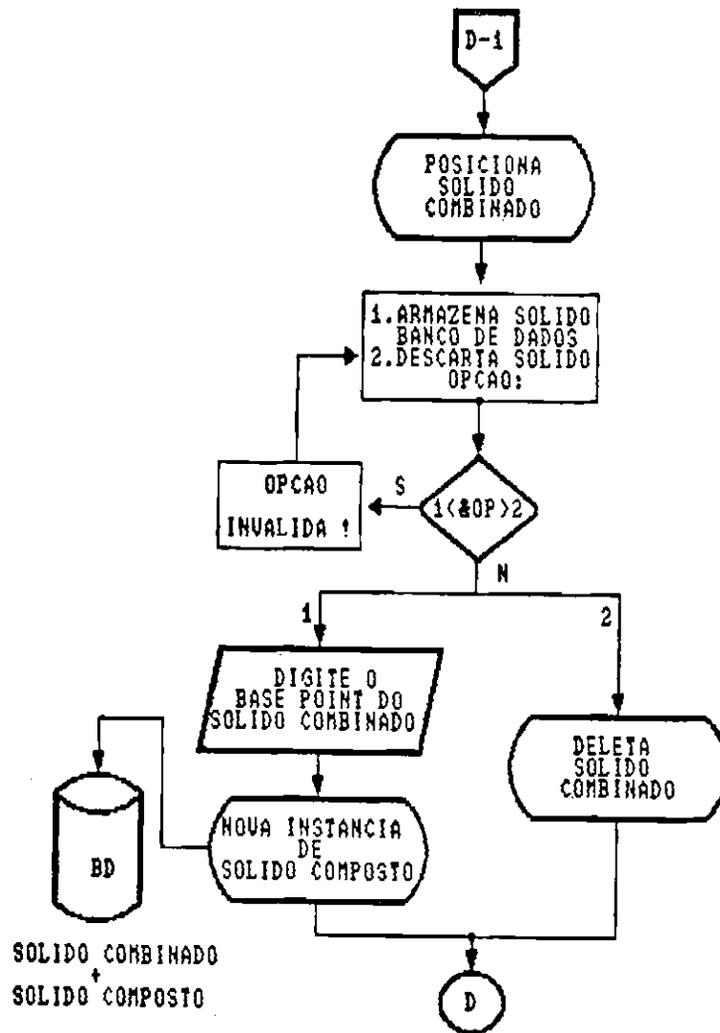


Fig. 6.5 - Continuação do fluxograma "sub-rotina sólido combinado".

Nesta sub-rotina estão presentes todas as ferramentas gráficas para as operações booleanas. A partir do menu pode-se escolher:

1. União

União de dois sólidos.

2. Interseção

Interseção entre dois sólidos.

3. Diferença

Subtrair um sólido de outro sólido.

4. Exit

Retornar ao menu operações com sólidos.

É solicitado ao usuário identificar ambos os sólidos usando a caneta de luz, assim como digitar a posição onde será colocado o sólido resultante. Em seguida, o sólido resultante é posicionado e dependendo do resultado ser satisfatório ou não, o sólido pode ser armazenado ou não, no banco de dados.

Caso seja armazenado, é solicitado digitar no sólido um ponto de referência, ou ponto de base. Este ponto é usado como referência para manipular o sólido em futuras operações. Uma nova instância de sólido composto é criada e ambos os sólidos são armazenados no banco de dados.

Se o sólido não for armazenado, este é apagado do modelo e o usuário pode retornar ao menu sólido combinado e tentar novamente a operação.

6.3.5 - FLUXOGRAMA "SUB-ROTINA SÓLIDO TRANSFORMADO"

Este fluxograma e sua continuação, estão mostrados nas figuras (Fig. 6.6) e (Fig. 6.7) a seguir.

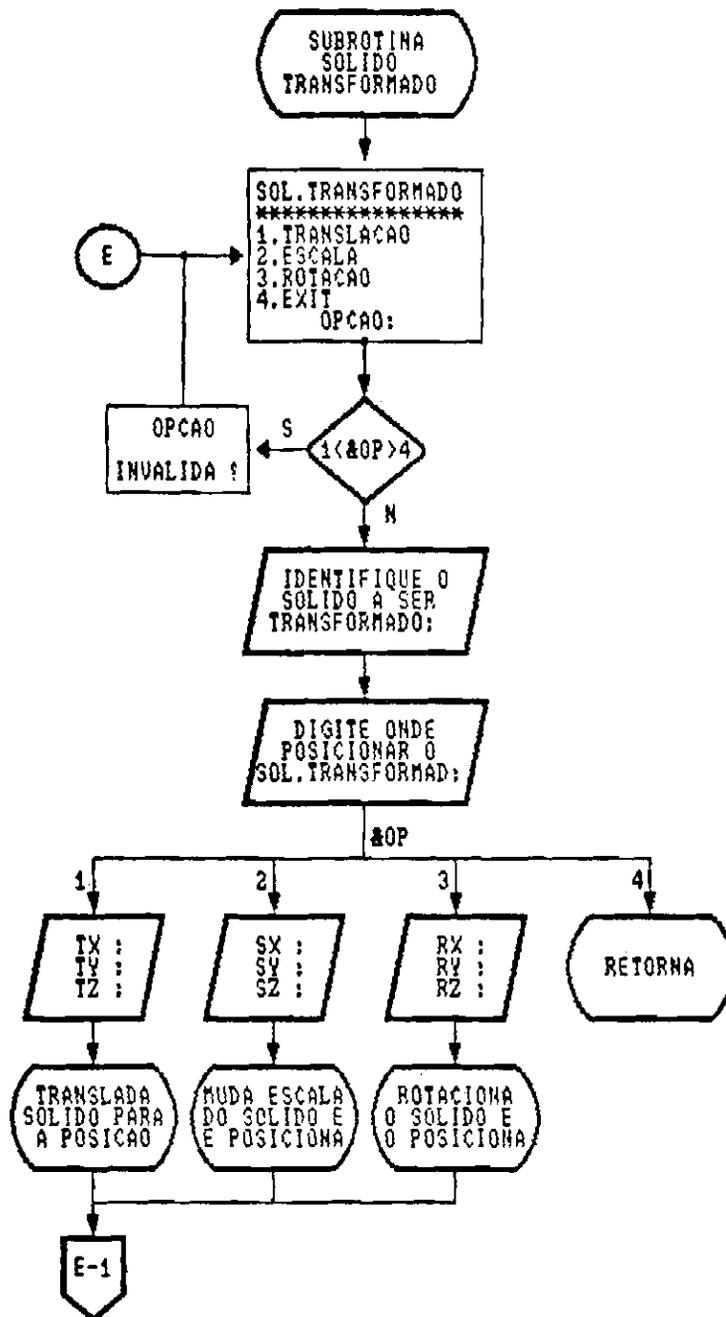


Fig. 6.6 - Fluxograma "sub-rotina sólido transformado".

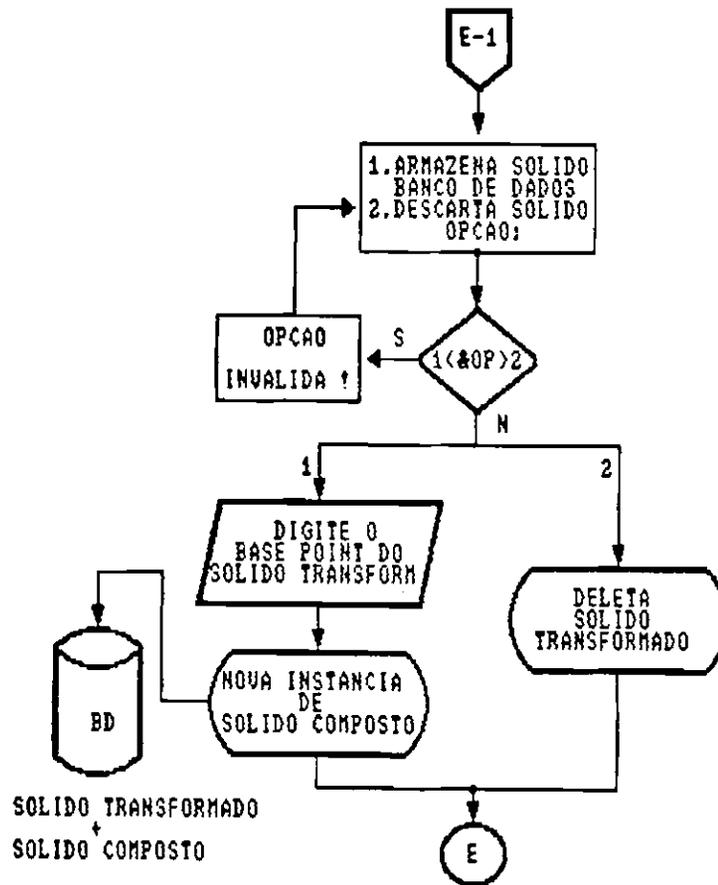


Fig. 6.7 - Continuação do fluxograma "sub-rotina sólido transformado".

Nesta sub-rotina estão implementadas as operações de transformação de corpo rígido. A partir do menu, o projetista ou usuário pode escolher:

1. Translação

Transladar um sólido. São solicitados os 3 parâmetros de translação. O sólido é transladado fazendo coincidir o ponto de referência do sólido com o ponto para onde este deve ser transladado.

2. Escala

Mudar o tamanho do sólido. Só está implementada a escala uniforme e não escalas diferentes para cada valor de coordenadas xyz do sólido; isto devido a uma das limitações do sistema. O sólido é mudado de escala tomando como referência, seu ponto de referência, anteriormente solicitado ao usuário.

3. Rotação

Rotacionar um sólido em torno dos eixos de coordenadas tomando como referência o ponto de referência do sólido. São solicitados os parâmetros de rotação.

4. Exit

Retorna ao menu operações com sólidos.

Assim, inicialmente é solicitado ao usuário identificar o sólido a ser transformado e digitar em que ponto do espaço o sólido resultante deve ser posicionado.

Uma vez posicionado o sólido transformado resultante, este pode ou não, ser armazenado no banco de dados. Se for armazenado, é solicitado um ponto de referência para este novo sólido, e o sólido transformado mais uma nova instância de sólido composto, são armazenados.

Caso contrário, o sólido é apagado e o usuário volta ao menu de sólido transformado e pode tentar novamente transformar o mesmo ou outro sólido.

6.3.6 - FLUXOGRAMA "SUB-ROTINA SÓLIDO PRIMITIVO"

Este fluxograma está mostrado na figura (Fig. 6.8) a seguir.

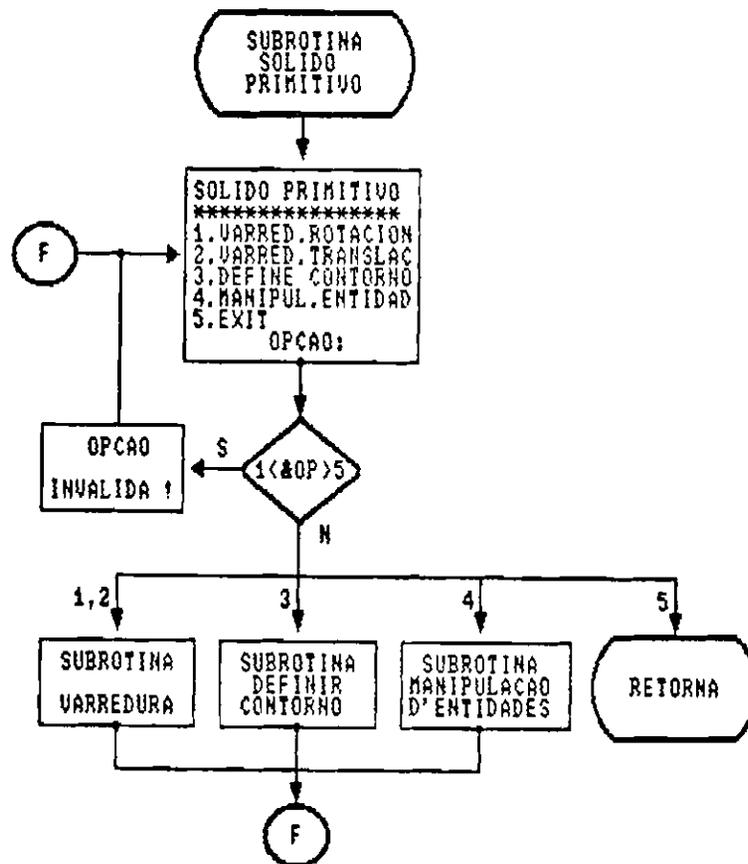


Fig. 6.8 - Fluxograma "sub-rotina sólido primitivo".

Esta sub-rotina permite construir um sólido primitivo. No menu existem as seguintes opções:

1. Varredura rotacional

Varre um contorno fechado em torno de um eixo de rotação e gera um sólido.

2. Varredura translacional

Varre um contorno fechado ao longo de um eixo de translação e gera um sólido.

3. Definir contorno

Permite construir, a partir de entidades 2D, um contorno fechado para ser utilizado pelas operações de varredura e determinar um sólido primitivo.

4. Manipulação de entidades 2D

Utilitário de apoio para a construção de um contorno e que apenas serve para entidades 2D. Permite mover, apagar, rotacionar e mudar de escalada uma entidade 2D.

5. Exit

Retorna ao menu de operações com sólidos.

6.3.7 - FLUXOGRAMA "SUB-ROTINA VARREDURA"

Este fluxograma está mostrado na figura (Fig. 6.9) a seguir.

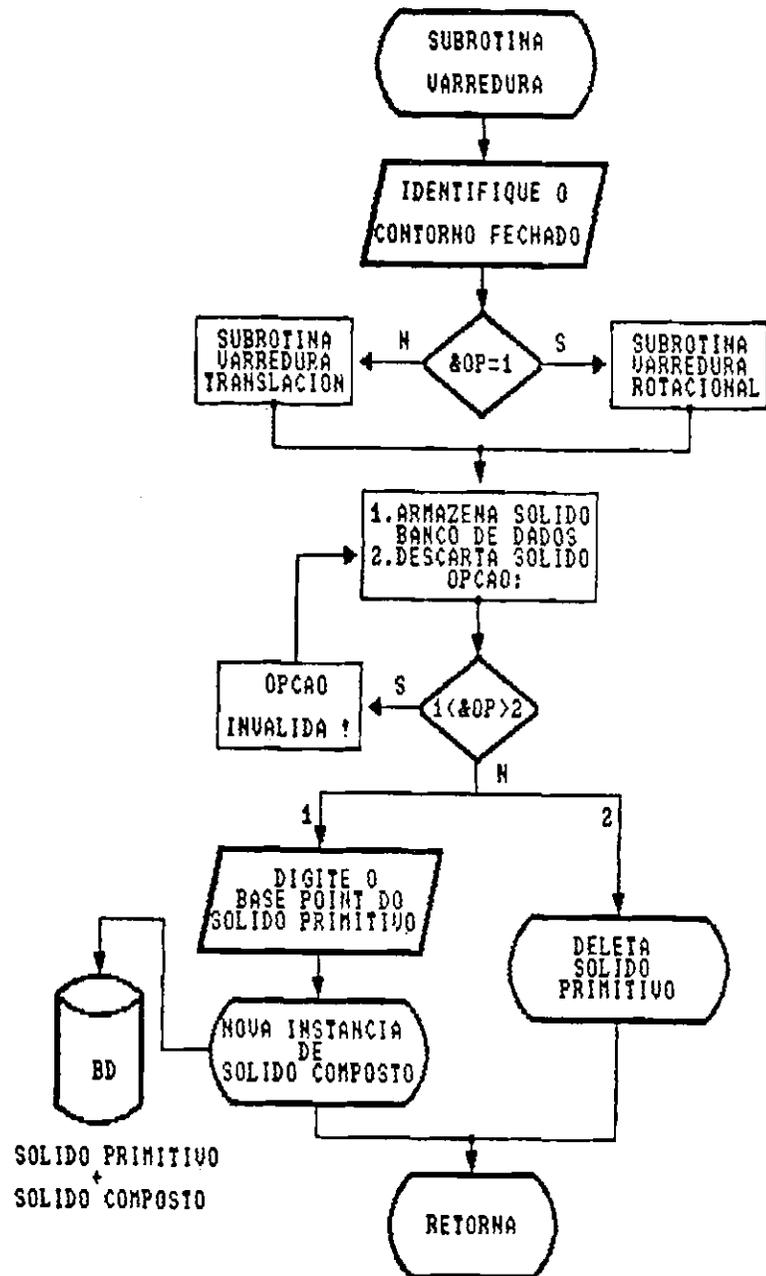


Fig. 6.9 - Fluxograma "sub-rotina varredura".

Esta sub-rotina engloba ambos os tipos de varredura.

Dependendo do tipo de varredura escolhido, é solicitado ao usuário identificar o contorno a ser varrido. O programa identifica o contorno e o varre gerando um sólido primitivo. Após o sólido ter sido gerado, este poderá ser armazenado ou não, no banco de dados.

Se o sólido resultante for armazenado, é solicitado ao usuário digitar um ponto de referência para este sólido. O sólido, assim como uma nova instância de sólido composto e a informação de cada uma das entidades que compõem o contorno fechado, são armazenados no banco de dados.

Caso contrário, o sólido é apagado e o programa retorna ao menu de sólido primitivo onde é possível varrer outro contorno.

6.3.8 - FLUXOGRAMA "SUB-ROTINA VARREDURA ROTACIONAL"

Este fluxograma está mostrado na figura (Fig. 6.10) a seguir.

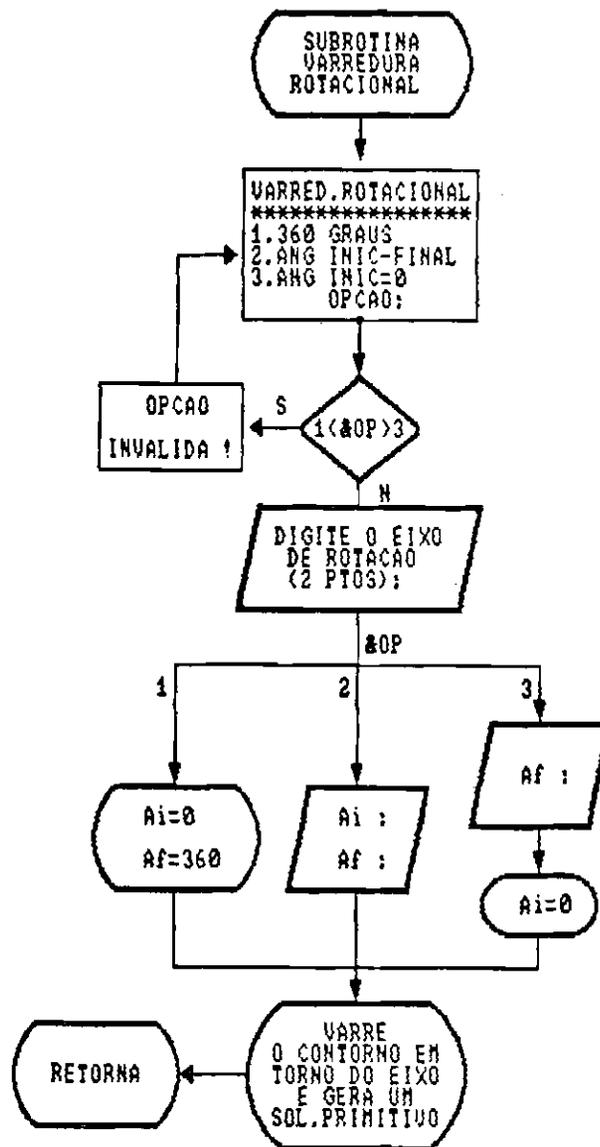


Fig. 6.10 - Fluxograma "sub-rotina varredura rotacional".

Esta sub-rotina varre rotacionalmente um contorno fechado e cria um sólido primitivo. No menu apresentado pode-se escolher:

1. 360 graus

Varredura completa em torno do eixo de rotação.

2. Ângulo inicial e ângulo final

Varredura parcial partindo do ângulo inicial até o ângulo final.

3. Ângulo inicial= 0

Varredura parcial a partir de 0 graus. Solicita-se ao usuário o valor do ângulo final.

Após a escolha do tipo de varredura, o usuário deve digitar os dois pontos que definem o eixo de rotação. Após a determinação do eixo, o contorno é varrido em torno do eixo escolhido, e um sólido primitivo é gerado.

É importante ressaltar que o eixo deve estar contido no mesmo plano do contorno fechado e este não poderá atravessar o contorno em hipótese alguma; no pior dos casos poderá estar contido em uma das entidades do contorno se esta entidade for uma reta, é claro.

6.3.9 - FLUXOGRAMA "SUB-ROTINA VARREDURA TRANSLACIONAL"

Este fluxograma está mostrado na figura (Fig. 6.11) a seguir.

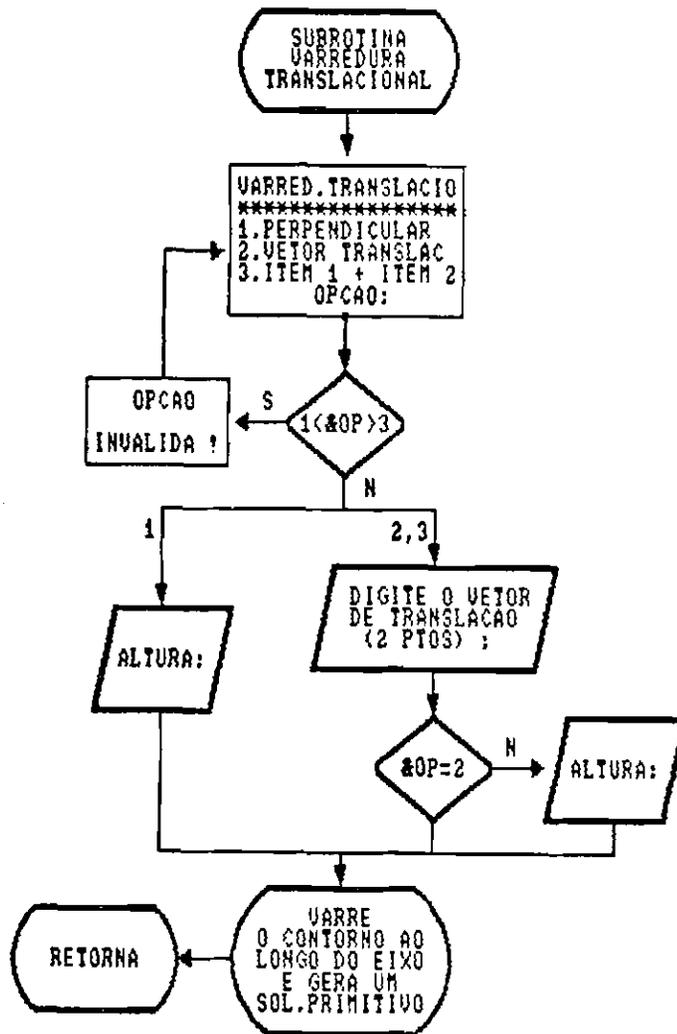


Fig. 6.11 - Fluxograma "sub-rotina varredura translacional".

Esta sub-rotina varre translacionalmente um contorno fechado e cria um sólido primitivo. No menu apresentado existem as seguintes opções:

1. Varredura perpendicular

Varredura perpendicular ao plano do contorno. Solicita-se a altura do sólido.

2. Vetor de translação

O usuário pode digitar 2 pontos definindo um vetor de translação. O módulo do vetor será a altura do sólido e a direção do vetor determina a direção do eixo de translação.

3. Item 1 + item 2

O usuário pode definir um vetor do qual é extraída a direção do vetor de translação, porém a altura do sólido deve ser fornecida via teclado.

Após a escolha do tipo de varredura, o contorno é varrido ao longo do eixo de translação e um novo sólido primitivo é gerado.

É importante ressaltar, que o eixo de translação não pode estar no mesmo plano do contorno em hipótese alguma. Se o valor da altura é um número negativo, o sólido é varrido para baixo.

Neste tipo de varredura, se for usado um vetor para definir a orientação da translação ou até mesmo a altura, é recomendável selecionar um plano de construção diferente do plano que contém o contorno, pois assim não haverá o risco de digitar o vetor no mesmo plano que o do contorno.

6.3.10 - FLUXOGRAMA "SUB-ROTINA DEFINIR CONTORNO"

Este fluxograma está mostrado na figura (Fig. 6.12) a seguir.

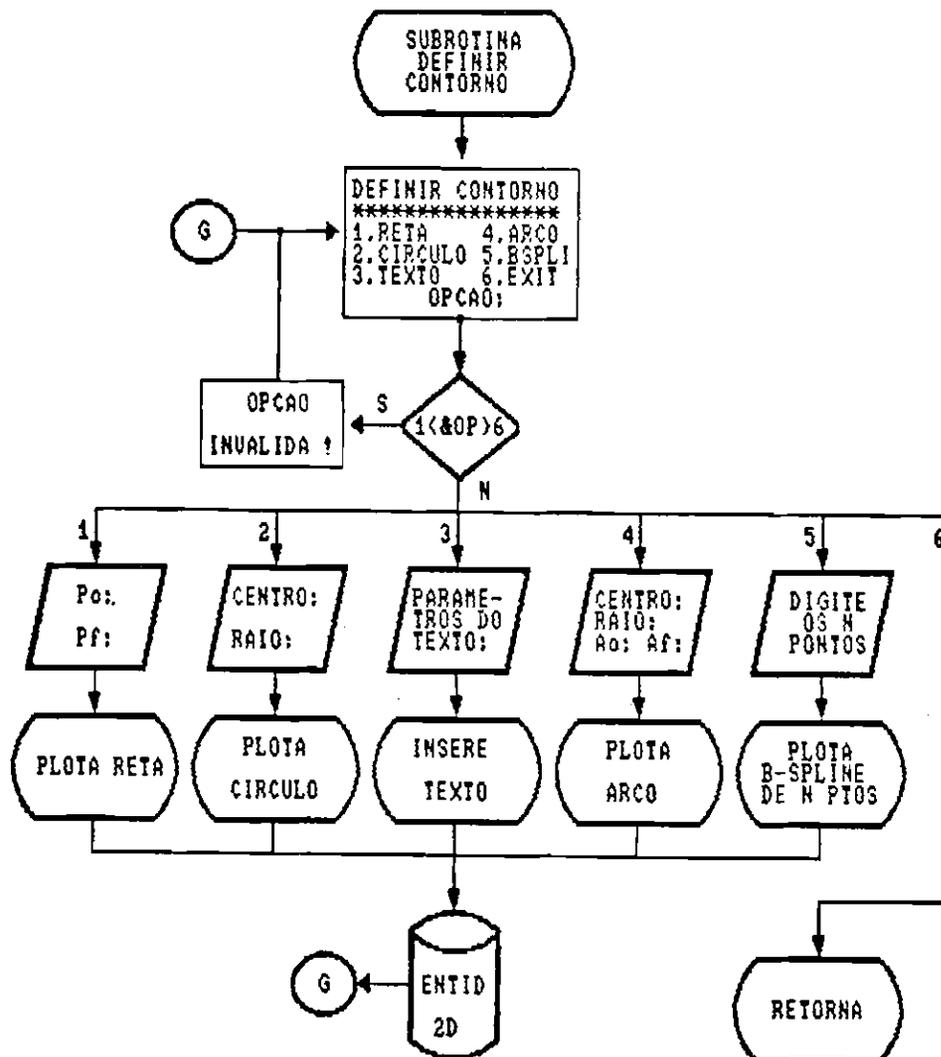


Fig. 6.12 - Fluxograma "sub-rotina definir contorno".

Esta sub-rotina gera um contorno fechado usando entidades 2D ou do nível 0. O menu apresentado, oferece:

1. Reta

Insere uma reta no modelo. Após o usuário digitar os pontos extremos da reta, a reta é exibida.

2. Círculo

Gera um círculo. Solicita-se o centro e o raio do círculo.

3. Texto

Insere um determinado tipo de texto na folha de desenho. São solicitados: altura do texto, inclinação, tipo de justificação, largura dos caracteres, nível de trabalho onde o texto será colocado e espessura dos caracteres.

4. Arco

Gera um arco. São solicitados o centro, raio, ângulo inicial e ângulo final. Estes parâmetros podem ser fornecidos via teclado ou via caneta de luz.

5. BSpline

Insere um curva B-Spline no modelo. É solicitado ao usuário digitar n pontos os quais são interpolados pela curva.

6. Exit

Retorna ao menu sólido primitivo.

Cada entidade construída é armazenada no banco de dados. Na construção do contorno, deve-se ressaltar que o contorno deverá ser totalmente fechado, isto é, todas suas entidades participantes devem estar interligadas.

6.3.11 - FLUXOGRAMA "SUB-ROTINA MANIPULAÇÃO DE ENTIDADES 2D"

Esta sub-rotina oferece ferramentas para manipular as entidades que definem um contorno.

Este fluxograma está mostrado na figura (Fig. 6.13) a seguir.

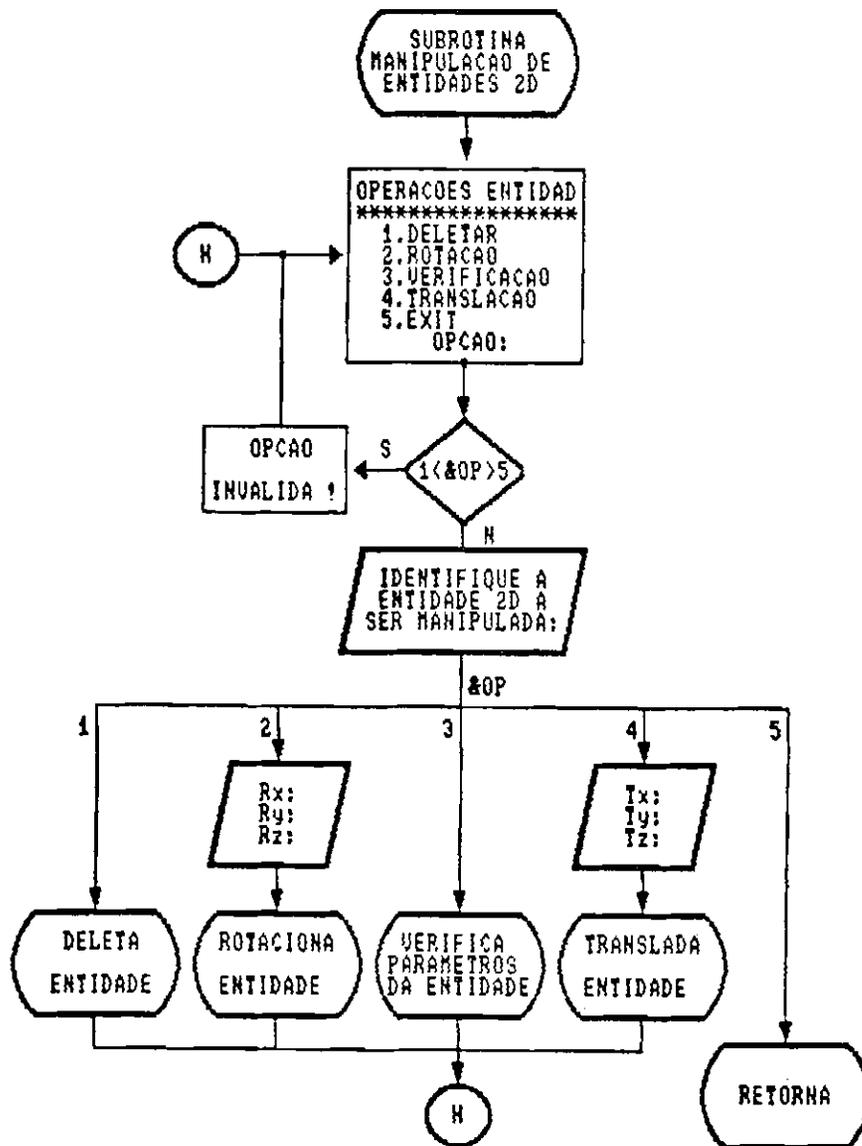


Fig. 6.13 - Fluxograma "sub-rotina manipulação de entidades 2D".

No menu apresentado pode-se escolher:

1. Deletar

Apaga uma entidade e a remove do banco de dados.

2. Rotação

Rotaciona uma entidade em torno dos eixos de coordenadas, dados os parâmetros de rotação e um ponto de referência na entidade. A informação da entidade é atualizada no banco de dados.

3. Verificação

Verifica os valores dos parâmetros de uma entidade. Por exemplo, as coordenadas dos pontos inicial e final de uma reta, o raio de um círculo, etc.

4. Translação

Translada uma entidade dados os parâmetros de translação solicitados. Também é solicitado um ponto de referência na entidade. A informação da entidade é atualizada no banco de dados.

5. Exit

Retorna ao menu sólido primitivo.

Em todos os casos, o usuário deverá identificar a entidade a ser manipulada usando a caneta de luz.

6.3.12 - FLUXOGRAMA "SUB-ROTINA SOMBREAMENTO"

Este fluxograma está mostrado na figura (Fig. 6.14) a seguir.

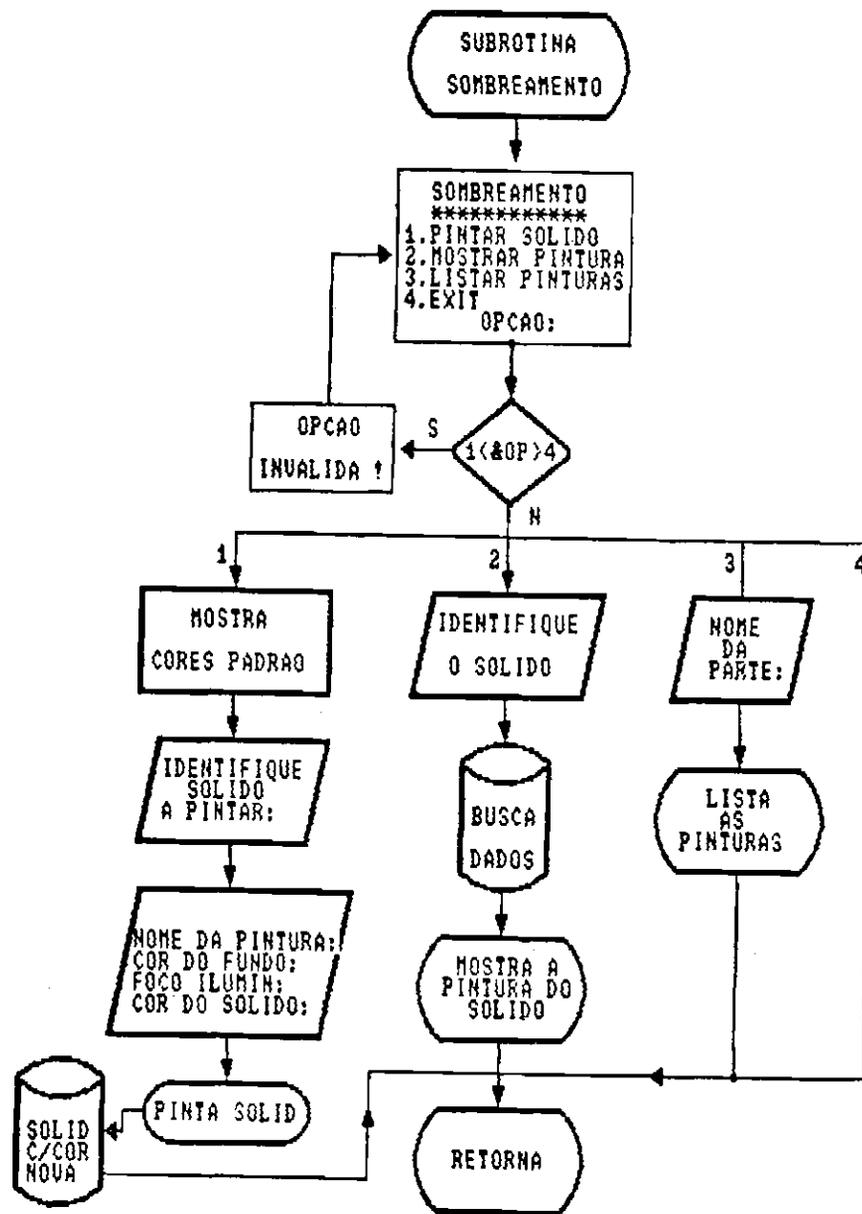


Fig. 6.14 - Fluxograma "sub-rotina sombreamento".

Esta sub-rotina permite definir características de cor e sombras no modelo. Do menu pode-se escolher:

1. Pintar sólido

Permite associar uma cor ao sólido.

Inicialmente é apresentado um menu de todas as cores disponíveis. O usuário deve identificar o sólido a ser pintado, fornecer um nome para a pintura ("picture") do sólido a ser criada, e escolher a cor do fundo, a cor que o sólido receberá e a posição do ponto de iluminação.

O programa busca no banco de dados a instância do sólido composto que representa este sólido e a batiza com o nome da pintura solicitada. Este nome é o nome da instância do sólido composto.

O sistema de coordenadas 3D do foco de iluminação está orientado com os eixos x e y coincidindo com o centro da tela de vídeo (0,0,0) e o eixo z na direção do tubo de raios catódicos. De tal maneira que a posição (0,0,1) colocaria o foco de luz no nariz do observador e um valor (1,1,1) no canto superior direito em frente a tela.

2. Mostrar pintura

Exibe a pintura de um sólido. O usuário apenas deve identificar com a caneta de luz o sólido. O programa busca no banco de dados a instância de sólido composto que aponta a este sólido, extrai o nome da instância, que corresponde ao nome da pintura, e mostra a pintura.

3. Listar pinturas

Lista o nome das pinturas de todos os sólidos que compõem o modelo de um arquivo de trabalho gráfico. O usuário deve apenas entrar o nome deste arquivo. Isto é útil para evitar dar o mesmo nome a vários sólidos.

4. Exit

Retorna ao menu operações com sólidos.

6.3.13 - FLUXOGRAMA "SUB-ROTINA DEFINIR PARÂMETROS"

Este fluxograma está mostrado na figura (Fig. 6.15) a seguir.

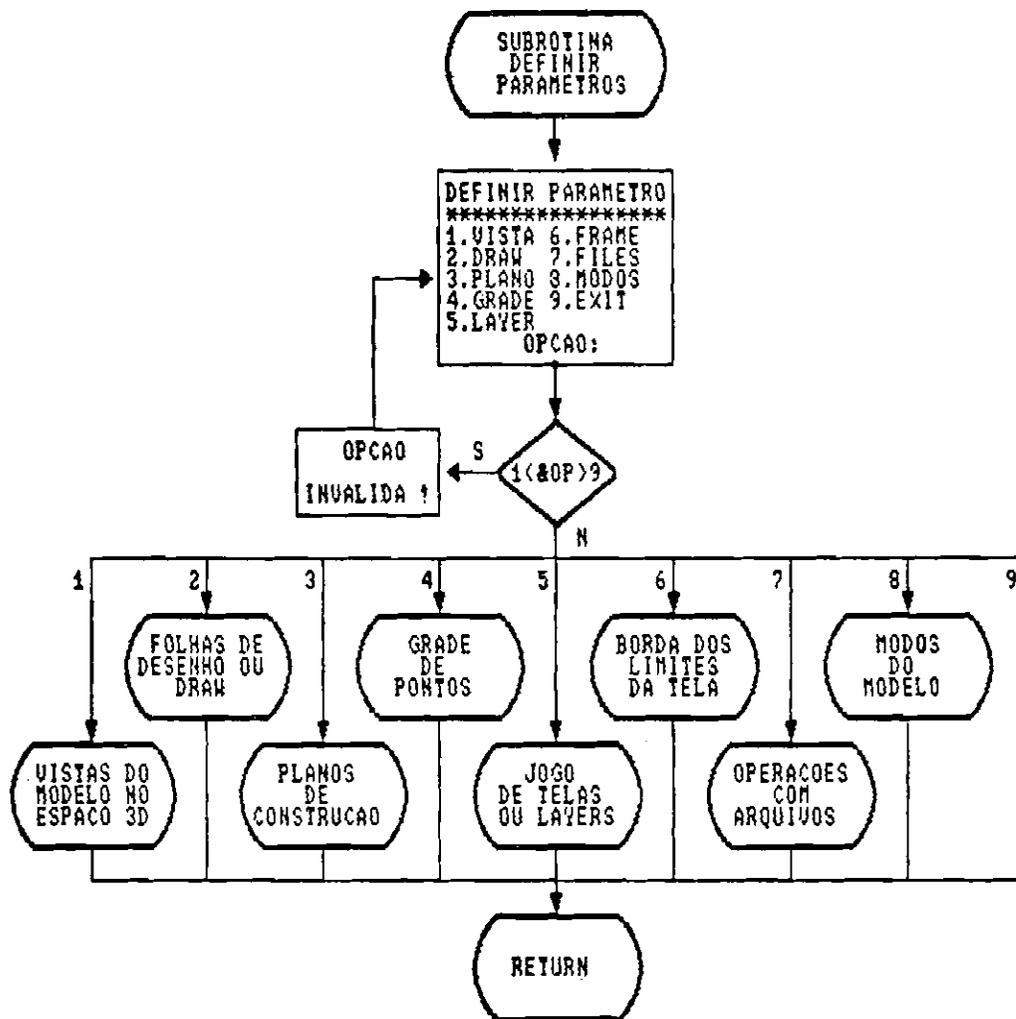


Fig. 6.15 - Fluxograma "sub-rotina definir parâmetros".

Esta sub-rotina permite definir o ambiente gráfico do modelo. No menu apresentado pode-se ver:

1. Vistas

Permite definir, apagar, mudar, ampliar ou reduzir, setar e resetar uma vista. Também é possível listar as vistas associadas com cada folha de desenho do arquivo de trabalho.

2. Draw

Permite ativar, apagar, ampliar ou reduzir uma folha de desenho. Também podem ser listadas todas as folhas de desenho associadas com um arquivo de trabalho.

3. Plano

Permite selecionar e definir planos de construção. Lista os planos de construção associados com um arquivo de trabalho, assim como liga ou desliga os eixos de coordenadas que representam a origem do sistema de coordenadas.

4. Grade

Seleciona a distância entre os pontos da grade, rotaciona a grade, lista informações da grade, liga ou desliga a grade.

5. Layer

Seleciona um nível de trabalho, ativa ou desativa determinados níveis, associa uma cor a um nível, lista os níveis ativos e desativos.

6. Frame

Liga ou desliga uma borda na tela. A borda é um retângulo tracejado que limita uma vista.

7. Files

Permite apagar, renomear, copiar ou listar arquivos. Também pode-se salvar um arquivo de trabalho ou imprimir a tela gráfica no "plotter".

8. Modos

Ativa os modos Model ou Draw.

9. Exit

Retorna ao menu principal.

6.3.14 - FLUXOGRAMA "SUB-ROTINA BANCO DE DADOS"

Este fluxograma está mostrado na figura (Fig. 6.16) a seguir.

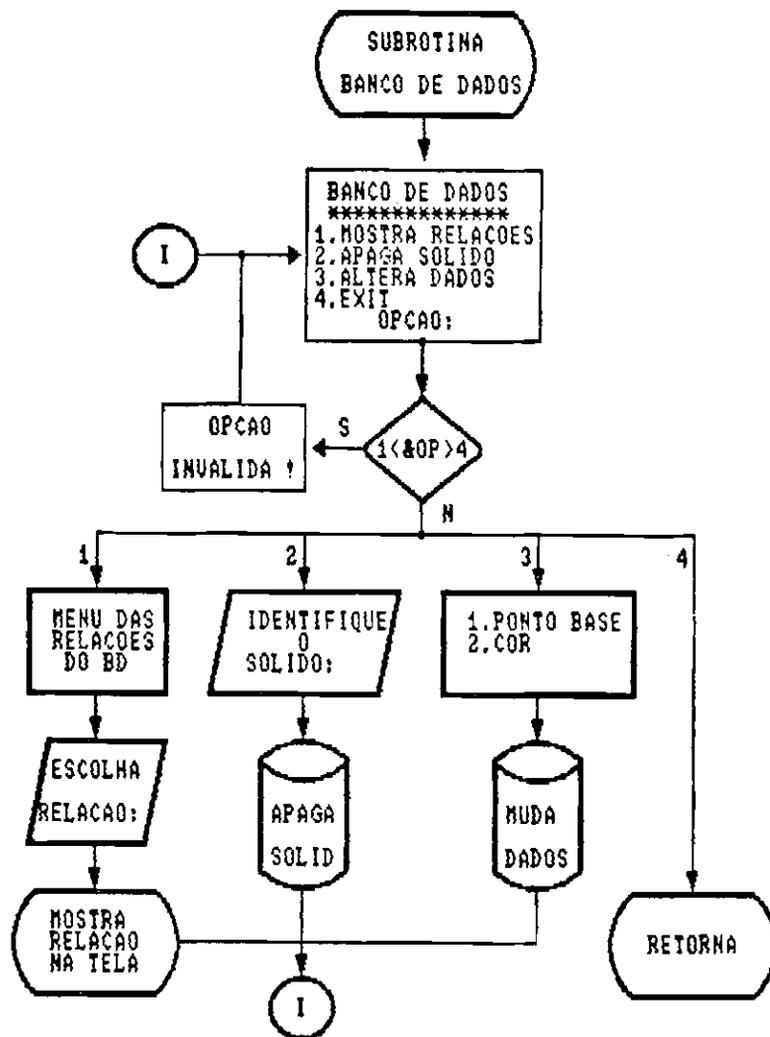


Fig. 6.16 - Fluxograma "sub-rotina banco de dados".

Esta sub-rotina permite manipular o banco de dados. No menu apresentado pode-se escolher:

1. Mostra relações

Mostra no vídeo cada uma das relações do banco de dados em um determinado momento durante o processo da modelagem de um sólido.

É apresentado um menu com todas as relações e o usuário escolhe uma. As relações são:

- sólido composto
- sólido combinado
- sólido transformado
- sólido primitivo
- contorno
- reta
- arco
- círculo
- B-Spline
- texto
- pontos

2. Apaga sólido

Permite apagar um sólido do banco de dados. É solicitado ao usuário identificar o sólido usando a caneta de luz.

O programa procura o sólido no banco de dados e apaga todos seus filhos dependentes. Como a estrutura de dados é hierárquica, retirar um sólido da estrutura, significa que se este não representa uma folha da árvore só pode ser um nó e portanto, deve-se eliminar a subárvore da qual este nó é pai. Portanto, todos seus dependentes são eliminados.

3. Alterar dados

Permite alterar o ponto de referência de um sólido assim como a pintura associada a este. É necessário que o usuário identifique o sólido. O programa procura o sólido no banco de dados e altera os dados.

CAPÍTULO 7

CONCLUSÕES E RECOMENDAÇÕES

Deixa-se implantado um programa de aplicação que implementa uma estrutura de dados hierárquica para modelamento de sólidos, o qual pode ser usado por qualquer usuário do sistema que deseje modelar um determinado sólido a partir da composição de sólidos mais simples, usando as operações de combinação e transformação, e com elementos em níveis intermediários gerados por varredura ("sweeping"), se desejado.

O programa de aplicação desenvolvido, que representa um modelador geométrico, assim como uma interface de alto nível entre o usuário e o sistema, fica guardado em disco à disposição dos usuários. Uma cópia fica em fita magnética.

Este programa de aplicação, desenvolvido na linguagem VARPRO 2, só pode ser executado no sistema CAD utilizado, por usar rotinas gráficas do pacote gráfico CADDS 4X.

Devido à linguagem VARPRO 2, ser uma linguagem interpretada, a execução do programa se tornava demasiada lenta devido a dois fatores: primeiro, todas as sub-rotinas implementadas deviam estar dentro do mesmo módulo de onde eram chamadas, o que fazia crescer o código fonte e, segundo, porque a cada chamada de uma sub-rotina, todo o código da sub-rotina era inserido no lugar da chamada.

A solução a este problema, que incrementou visivelmente a velocidade de execução, embora o programa seja lento por si mesmo devido ao sistema, foi quebrar o programa principal em vários pequenos programas, ou módulos.

Na verdade cada sub-rotina se tornou um módulo independente, controlados por um módulo central que é o programa principal. Assim, o programa principal ao invés de chamar uma sub-rotina, executa outro programa e quando este segundo programa finaliza, retorna ao programa principal. E nada impede que um programa executado, execute outros programas ou módulos, como é o que realmente acontece.

A própria estrutura de dados manipulada, que define um processo de construção de um sólido seguindo uma árvore binária, determina que estes processos de execução de módulos independentes também sigam uma sequência de chamadas ordenadas como uma árvore binária.

Com esta independência de módulos, além de ganhar em velocidade, torna-se a depuração eficaz pois cada módulo é independente. Achar os erros lógicos e de sintaxe torna-se mais fácil e veloz.

Na época da implementação do modelador geométrico, como usuário do sistema, foi-me otorgado o subdiretório de trabalho *SYSTEM.JAD.ANDRESRUN, portanto, todos os módulos que compõem o programa principal do modelador, se encontram neste subdiretório.

Cada módulo é um arquivo texto que contém instruções na linguagem VARPRO 2. Portanto, todos os fontes dos módulos implementados estão à disposição para serem analisados, avaliados ou até melhorados por quem venha a se interessar em dar continuidade a este trabalho. Todos os nomes destes módulos, que na verdade são arquivos texto, têm o prefixo *SYSTEM.JAD.ANDRESRUN.&BCD.

O prefixo *SYSTEM.JAD.ANDRESRUN indica o subdiretório, e o prefixo &BCD indica que o arquivo é do tipo texto. Na verdade o prefixo que indica o subdiretório é arbitrário.

O banco de dados gerado pela aplicação, também é guardado como um arquivo texto e o nome deste arquivo é formado pelo prefixo do subdiretório, seguido do nome do arquivo de trabalho do usuário e com o sufixo .BD .

Todas as pinturas dos sólidos criadas durante a execução do programa, também são gravadas no disco como arquivos com formato de desenho especiais e cujos nomes levam o prefixo *SYSTEM.JAD.ANDRESRUN.&PICT .

Como foi mencionado anteriormente, o fator que torna o programa de aplicação muito lento, e que também representa um obstáculo no desenvolvimento, é o fato do processador do sistema que controla toda a atividade multiusuária ser de 16 bits o que determina que o acesso às partes, assim como o armazenamento destas e as chamadas aos arquivos de trabalho, se tornem muito lentas.

Este é um fator que geralmente limita ao usuário em função do tempo de resposta do sistema. Este limite é em virtude da generalidade de aplicações do sistema, que também é capaz de trabalhar em outras áreas como elementos finitos, esquemas eletrônicos, projetos de circuito impresso, instalações elétricas, etc.

Dentro do contexto do trabalho desenvolvido, o modelador geométrico permite modelar o protótipo de um projeto a ser executado. Por exemplo, o projeto de um satélite, e um modelo de seu protótipo no mundo real, pelas facilidades oferecidas, fornecendo uma visão real antecipada de como será o projeto final.

Assim, a flexibilidade do modelador geométrico de alterar a estrutura do modelo, permite mudar aspectos do protótipo antes não percebidos, pela incapacidade de uma simulação visual no mundo real, de tal maneira a ter liberdade de conceber vários modelos finais de um mesmo projeto, e escolher o mais adequado.

A documentação existente para o sistema é bem detalhada pois existem vários manuais que atendem todos os recursos disponíveis.

Para o contexto do trabalho, recomenda-se os manuais citados na bibliografia anexa ao final deste trabalho, e que abordam o sistema operacional, a linguagem VARPRO 2, e o pacote gráfico CADD5 4x que descreve detalhadamente as entidades geométricas, definição de vistas, níveis de trabalho, planos de construção, transformações, operações booleanas, sombreado, e outras informações, o que certamente, permitirá que a pessoa interessada em dar continuidade a este trabalho possa compreender como funciona o software que implementa este modelador geométrico.

Porém a versão de CADD5 4X usada na implantação, não possui a documentação completa dos recursos disponíveis, tais como para o caso das operações booleanas. Em certos casos críticos estas não se realizam e não há explicações detalhadas nos manuais para identificar as causas. Isto representa uma limitação, pois impossibilita a implementação de testes de validação antes da execução destas funções, já que não é possível identificar e descartar os casos críticos.

Acredita-se que novas versões do modelo superem estas deficiências, o que também implicará na necessidade de uma manutenção de software no programa de aplicação desenvolvido, visando sua atualização conforme a evolução das versões.

Surgirão portanto alguns problemas para os usuários que venham utilizar este modelador geométrico. Isto só poderá ser resolvido quando a versão do software disponível do pacote gráfico CADD5 4X for atualizada e melhorada.

Outro problema é o fato da linguagem VARPRO 2 ser exclusiva do sistema o que dificulta a transportabilidade do software a outro sistema. Entretanto, a estrutura do banco de dados foi apresentada e assim, como foi possível ser implantado neste sistema, poderia ser implantado em outro sistema CAD.

A partir da estrutura de dados hierárquica e a estrutura lógica do programa de aplicação (detalhada através dos fluxogramas anteriores), apresentados neste trabalho, pode ser possível a implementação e transportabilidade do modelador geométrico a outros sistemas CAD.

Uma etapa posterior a este trabalho seria justamente a de desenvolver a transportabilidade do software, mantendo-lhe as características de hierarquia e de utilização de técnicas de varredura, além da estrutura do banco de dados aqui mostrado.

Este objetivo, poderia ser atingido usando pacotes gráficos tridimensionais padronizados, como é o caso dos padrões GKS-3D e PHIGS, e escrevendo o programa de aplicação, que implementa o modelador geométrico, em uma linguagem de alto nível suportada por estes padrões, como é o caso das linguagens C, Pascal e FORTRAN.

Este trabalho também vem demonstrar, pela experiência adquirida, que dado um sistema CAD com uma série de manuais de documentação, e a partir de um estudo profundo do funcionamento do sistema e avaliação de toda sua capacidade gráfica, é possível implementar programas de aplicação visando atender determinados objetivos, que representem uma interface entre o usuário e o software gráfico disponível no sistema, de tal maneira que se facilite a utilização pelo usuário, de forma que este não precise se preocupar com detalhes técnicos, ou até mesmo com programação, e atinja seus objetivos desejados.

Esta interface deve ser implementada através de menus de tela e o diálogo deve ser claro e amigável ao usuário, evitando o uso de termos técnicos que podem confundir-lo.

Estes menus de tela, devem ser conduzidos com um número de opções limitado de tal maneira que o usuário não tenha condições de errar. Assim mesmo, para as solicitação de entrada de dados pelo usuário, devem ser emitidas mensagens claras e até mesmo detalhadas, como por exemplo, solicitar usar aspas duplas quando a entrada é um dado alfanumérico. Isto favorece tanto ao usuário como ao implementador, em razão de minimizar os erros devido à sintaxe e facilitar a operação pelo usuário.

REFERÊNCIAS BIBLIOGRÁFICAS

MORTENSON M.E. Geometric modeling. New York, John Wiley, 1985.

NEWMAN W.M.; SPROULL R.F. Principles of interactive computer graphics. New York, McGraw-Hill, 1979.

CONTROL DATA CORPORATION. Cyber 910, "IRIS User Guide". New York, 1986.

REQUICHA A.A.G. Representation for rigid solids: Theory, methods and systems. ACM Association for Computing Machinery, 12(4):437-464, Dec. 1980.

COMPUTERVISION CORPORATION. CADDS 4X CAD/CAM Concepts Book, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984.

APÊNDICE A

BIBLIOGRAFIA COMPLEMENTAR

- AGIN, G.J. Hierarquical representation of tree dimensional objects using verbal models. IEEE Transaction on Pattern Analysis in Machine Intelligence, 3(2):197-204, Mar. 1980.
- BETTELS, J.; BONO, P.R.; MCGINNIS, E.; RIX, J. Guidelines for determining when to use GKS and when to use PHIGS. Computer Graphics Forum, 7(4):323-329, Dec. 1988.
- BIERI, H.; NEF, W. A sweep-plane algorithm for computing the volume of polyhedra represented in boolean form. Linear Algebra and its Applications, 52(53):69-79, 1983.
- BONO, P.R. Guest editor's introduction to graphics standards. IEEE Computer Graphics and Applications, 6(8):12-16, Aug. 1986.
- BRODLIE, K.W. GKS programs - are they portable?. University of Leeds, s.d.
- CARSON, G.S.; MCGINNIS, E. The reference model for computer graphics. IEEE Computer Graphics and Applications, 8(6):17-23, Aug. 1985.
- CODD, E.F. Extending the database relational model to capture more meaning. ACM Transactions on Database System, 4(4):397-434, Dec. 1979.

A.2

COMPUTERVISION CORPORATION. CGOS 200X Operator Guide Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984a.

CADDS 4X CAD/CAM Concepts Book, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984b.

Introduction to CADDS 4X Commands, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984c.

CADDS 4X CADDS Access Reference, Designer V-X and CDS 4000 Software Documentation Summaries. July 1984d.

CADDS 4X Command Processor Reference, Designer V-X and CDS 4000 Software Documentation Summaries. July 1984e.

CADDS 4X Part and Drawing Environment Reference Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984f.

CADDS 4X Basic Geometry/Graphics Construction Reference, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984g.

CADDS 4X Basic Geometry/Graphics Editing 1 Reference, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984h.

CADDS 4X Basic Geometry/Graphics Editing 2 Reference, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984i.

A.3

COMPUTERVISION CORPORATION. CADDS 4X Drafting Projection User Guide, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984j.

CADDS 4X Appearance Reference, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984k.

CADDS 4X Text and Text File Manipulation Reference Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984l.

CADDS 4X Figures Reference, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984m.

CADDS 4X Menuing Reference, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984n.

CADDS 4X Surface Reference, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984o.

CADDS 4X Bezier Geometry Reference, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984p.

CADDS 4X Subassembly Reference, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984q.

A.4

COMPUTERVISION CORPORATION. CADDS 4X Data Base Maintenance Reference, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984r.

CADDS 4X VARPRO 2 Language Reference, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984s.

CADDS 4X Advanced Surface Design Reference Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984t.

CADDS 4X 3D Mechanical Design Workbook, Designer-V X and CDS 4000 Software Documentation Summaries. New York, July 1984u.

CDS 4000 GET/PUT Part Reference, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984v.

CDS 4000 32-Bit Processor Graphics Display Facility, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984w.

CDS 4000 CV-Pascal User Guide, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984x.

CDS 4000 Soliddesign Reference, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984y.

CDS 4000 Imagedesign Reference, Designer V-X and CDS 4000 Software Documentation Summaries. New York, July 1984z.

CONTROL DATA CORPORATION. Cyber 910, "IRIS User Guide".
New York, 1986a.

Cyber 910, "IRIS Programming Tutorial". New York,
1986b.

COSSU, R.; ERCOLLI, M.; MOLTEDO, L. An extension of CGI
for generation and manipulation of raster images. IEEE
Computer Graphics and Applications, 13(1):39-48, 1989.

DATE, C.J. Introdução a sistemas de banco de dados. Rio
de Janeiro, Campus, 1984.

DIAZ, G.A. Project no.3 B-Splines surface fitting. São
José dos Campos, INPE, Mar. 1983.

FOLEY, J.D.; VAN DAM, A. Fundamentals of interactive
computer graphics. New York, Addison-Wesley, Mar. 1982.

FOLEY, J.D.; WENNER, P.A. Core system implementation. ACM
Computer Graphics and Applications, 15(3):123-131, Aug.
1981.

GOEBEL, M.; KROENKER, D. A multi-microprocessor GKS
workstation. IEEE Computer Graphics and Applications,
6(7):54-60, July 1986.

HARRINGTON, S. Computer graphics: A programming approach.
New York, McGraw-Hill, 1983.

HENDERSON, L.; JOURNEY, M.; OSLAND, C. The computer
graphics metafile. IEEE Computer Graphics and
Applications, 6(8):24-32, Aug. 1986.

A.6

- HERTEL, S.; MANTILA, M.; MEHLHORN, K.; NIEVERGELT, J.
Space sweep solves intersection of convex polyhedra.
Acta Informática, 21(5):501-519, Aug. 1984.
- KAWAGOE, K.; MANAGAKI, M. Parametric object model: A
geometric data model for Computer-aided-engineering.
C & C System Research Laboratories, Sept. 1983.
- KETABCHI, M.A.; BERZINS, V. Modeling and managing CAD
databases. IEEE Computer Graphics and Applications,
20(2):93-102, Feb. 1987.
- KETABCHI, M.A.; BERZINS, V.; MARCH, S. An object-oriented
data model for design databases. In: COMPUTER SCIENCE
CONFERENCE, 1986. Proceedings.
- LITTLE, C.T. Graphics and GKS at the UK Metereological
Office. U.K. Metereological Office Berkshire, Bracknell,
s.d.
- LONGHI, M.T.; FREITAS, C.M.; GOLENDZINER, L.G. O Modelo
geométrico do N-MOS/UFRGS. In: SIMPÓSIO BRASILEIRO DE
BANCO DE DADOS, 2., Porto Alegre, 7-8 maio 1987, Anais.
Porto Alegre, UFRS, 1987, p. 11-21.
- MAGNENAT, N.; THALMANN, D.T. Advanced course on image
synthesis and computer animation. In: COMPUTER SCIENCE
SCHOOL, 6., Campinas, 1988.
- MIRANTE, A.; WEINGATTEN, N. The radial sweep algorithm for
constructing triangulated irregular networks. IEEE
Computer Graphics and Applications, 2(5):11-21, May 1982.

A.7

- NIEVERGELT, J.; PREPARATA, F.P. Plane sweep algorithms for intersecting geometric figures. Communications of the ACM, 25(10):739-747, Oct. 1982.
- PERSIANO, R.C.M.; OLIVEIRA, A.A.F. Introdução à computação gráfica. Belo Horizonte. Publicado para a V Escola de Computação, 1986.
- PFAFF, G.E. Evolution of graphics standards. Darmstadt, s.ed., s.d.
- POWERS, T.; FRANKEL, A.; ARNOLD, D. The computer graphics virtual device interface. IEEE Computer Graphics and Applications, 6(8):33-41, Aug. 1986.
- PUK, R.F.; MCCONNELL, J.I. GKS-3D: A three-dimensional extension to the graphical kernel system. IEEE Computer Graphics and Applications, 6(8):42-49, Aug. 1986.
- REQUICHA, A.A.G. Mathematical models of rigid solid objects. Univ. Rochester, Nov. 1977. (Production Automation Project TM, 28).
- REQUICHA, A.A.G. Representations of rigid solid objects. In: ENCARNAÇÃO J., ed. Computer aided design, lecture notes on computer science. New York, Springer-Verlag, 1980, p. 89.
- REQUICHA, A.A.G.; VOELCKER, H.B. Constructive solid geometry. Univ. Rochester, Nov. 1977. (Production Automation Project TM, 25).

- REQUICHA, A.A.G.; TILOVE, R.B. Mathematical foundations of constructive solid geometry: General topology of regular closed sets. Univ. Rochester, Nov. 1978. (Production Automation Project TM, 28).
- ROGERS, D.F. Procedural elements for computer graphics. New York, McGraw-Hill, 1986.
- ROSENTHAL, D.S.H.; MICHENER, J.C.; PFAFF, G.; KESSENER, R.; SABIN, M. The detailed semantics of graphics input devices. ACM Computer Graphics and Applications, 11(3):33-38, July 1982.
- SCHOENHUT, J. Are PHIGS and GKS necessarily incompatible?. IEEE Computer Graphics and Applications, 6(8):51-53, July 1986.
- SETZER, V.W. Projeto lógico e projeto físico de banco de dados. Belo Horizonte, 1986. Publicado para a V Escola de Computação.
- SHUEY, D.; BAILEY, D.; MORRISSEY, T.P. PHIGS: A standard, dynamic, interactive graphics interface. IEEE Computer Graphics and Applications, 6(8):50-57, Aug. 1986.
- SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS (SIBGRAPI), 2., Águas de Lindóia, abr. 26-28, 1989. Anais. São José dos Campos, INPE, 1989.
- SIMONS, R.W. Minimal GKS. Computer Graphics and Applications, 7(3):183-189, July 1983.

- SMITH, J.M.; SMITH, D.C.P. Database abstractions: Aggregations. Communicatons of the ACM, 20(6):405-413, June 1977.
- _____ Database abstractions: aggregation and generalization. ACM Transactions on Database Systems, 2(2):105-133, June 1977.
- SPARKS, M.; GALLOP, J.R. Language bindings for computer graphics standards. IEEE Computer Graphics and Applications, 6(8):58-65, Aug. 1986.
- SPLERS, R.G. Realization and application of an intelligent GKS workstation. IEEE Computer Graphics and Applications, 6(5):58-65, May 1986.
- TILOVE, R.B.; REQUICHA, A.A.G. Closure of boolean operations on geometric entities. Compututer Aided Design, 12(5):219-220, Sept. 1980.
- ULLMAN, J. Principles of database systems. 2.ed., Rockville, MD, Computer Science, 1982. 484 p.
- VOELCKER, H.B.; REQUICHA, A.A.G. Geometric modelling of mechanical parts and processes. IEEE Computer Graphics and Applications, 10(12):48-57, Dec. 1977.
- WAGGONER, C.N.; TUCKER, C.; NELSON, C.J. NOVA GKS: A distributed implementation of the graphical kernel system. ACM Computer Graphics, 18(3):275-282, July 1984.

